

**PRIN. K.P. MANGALVEDHEKAR INSTITUTE OF MANAGEMENT  
CAREER DEVELOPMENT AND RESEARCH**

Approved by AICTE, Govt. Of India, Govt. of Maharashtra  
"Affiliated to Punyashlok Ahilyadevi Holkar Solapur University, Solapur"

BCA PPT



**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of**  
**Management Career Development and**  
**Research.**

**156-B Railway Lines Solapur.**  
**Affiliated PAH Solapur University**

Name of Faculty . Mr Santosh Kulkarni

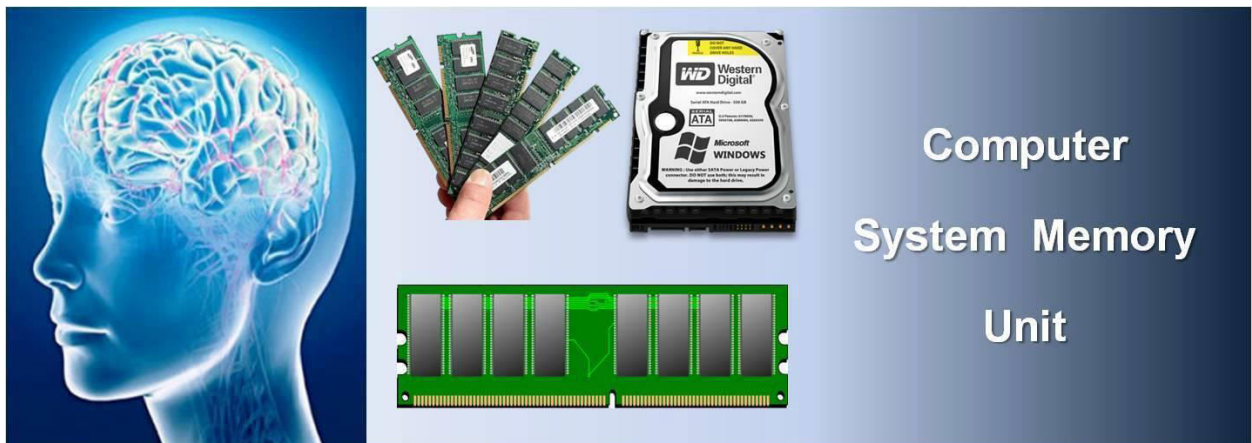
Subject Fundamentals of Computer  
Programme:- BCA I Sem I

Name of Faculty :-Santosh Kulkarni

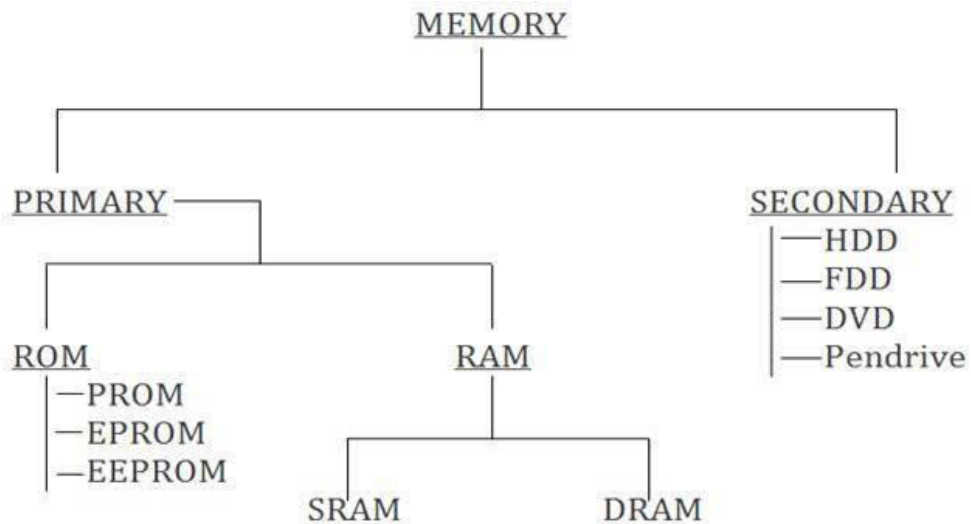
Subject :Fundamentals of Computer

Program:-BCA I SEM I

Computer Memory



**Computer memory** is divided into main (or primary) **memory** and auxiliary (or secondary) **memory**. Main **memory** holds instructions and data when a program is executing, while auxiliary **memory** holds data and programs not currently in use and provides long-term storage.



- **1. Random Access Memory (RAM) –**
- It is also called as *read write memory* or the *main memory* or the *primary memory*.
- The programs and data that the CPU requires during execution of a program are stored in this memory.
- It is a volatile memory as the data loses when the power is turned off.
- RAM is further classified into two types- *SRAM (Static Random Access Memory)* and *DRAM (Dynamic Random Access Memory)*.
  
- **Read Only Memory (ROM) –**
- Stores crucial information essential to operate the system, like the program essential to boot the computer.



- It is not volatile.
  - Always retains its data.
  - Used in embedded systems or where the programming needs no change.
  - Used in calculators and peripheral devices.
  - ROM is further classified into 4 types- *ROM*, *PROM*, *EPROM*, and *EEPROM*.
- **Types of Read Only Memory (ROM) –**
    - **PROM (Programmable read-only memory)** – It can be programmed by user. Once programmed, the data and instructions in it cannot be changed.
    - **EPROM (Erasable Programmable read only memory)** – It can be reprogrammed. To erase data from it, expose it to ultra violet light. To reprogram it, erase all the previous data.
    - **EEPROM (Electrically erasable programmable read only memory)** – The data can be erased by applying electric field, no need of ultra violet light. We can erase only portions of the chip.

RAM	ROM
1. Temporary Storage.	1. Permanent storage.
2. Store data in MBs.	2. Store data in GBs.
3. Volatile.	3. Non-volatile.
4.Used in normal operations.	4. Used for startup process of computer.
5. Writing data is faster.	5. Writing data is slower.

### Difference between RAM and ROM

DRAM	SRAM
1. Constructed of tiny capacitors that leak electricity.	1. Constructed of circuits similar to D flip-flops.
2. Requires a recharge every few milliseconds to maintain its data.	2. Holds its contents as long as power is available.
3. Inexpensive.	3. Expensive.
4. Slower than SRAM.	4. Faster than DRAM.
5. Can store many bits per chip.	5. Can not store many bits per chip.
6. Uses less power.	6. Uses more power.
7. Generates less heat.	7. Generates more heat.
8. Used for main memory.	8. Used for cache.

### Difference between SRAM and DRAM

Secondary Storage Devices



- Magnetic Tape memory

- Magnetic drums, magnetic tape and magnetic disks are types of magnetic memory. These memories use property for magnetic memory. Here, we have explained about magnetic tape in brief.

- **Magnetic Tape memory :**

In magnetic tape only one side of the ribbon is used for storing data. It is sequential memory which contains thin plastic ribbon to store data and coated by magnetic oxide. Data read/write speed is slower because of sequential access. It is highly reliable which requires magnetic tape drive writing and reading data.

- **Advantages :**

- These are inexpensive, i.e., low cost memories.
- It provides backup or archival storage.
- It can be used for large files.
- It can be used for copying from disk files.
- It is a reusable memory.
- It is compact and easy to store on racks.

- **Disadvantages :**

- Sequential access is the disadvantage, means it does not allow access randomly or directly.
- It requires caring to store, i.e., vulnerable humidity, dust free, and suitable environment.
- It stored data cannot be easily updated or modified, i.e., difficult to make updates on data.
- A **magnetic disk** is a storage device that uses a magnetization process to read, write, rewrite and access data. It is covered with a **magnetic** coating and stores data in the form of tracks, spots and sectors. Hard **disks**, zip **disks** and floppy **disks** are common examples of **magnetic disks**.
- **Magnetic disks** are the most common backing storage device. **Data** is **stored** using magnetised spots called domains on the **disk**. Each domain can store one bit of **data** (a 0 or a 1). ... **Data** is read from the **disk** using a **disk** head which moves mechanically about the **disk** (rather like a record player tone arm).
- A **magnetic disk** is a storage device that uses a magnetization process to write, rewrite and access data. It is covered with a **magnetic** coating and stores data in the form of tracks, spots and sectors. Hard **disks**, zip **disks** and **floppy disks** are common **examples of magnetic disks**

- A compact disc is a portable **storage** medium that can be used to record, store and play back audio, video and other data in digital form.
- Most of a **CD** is **made from** a tough, brittle plastic called polycarbonate. Sandwiched in the middle there is a thin layer of aluminum. Finally, on top of the aluminum, is a protective layer of plastic and lacquer. The first thing you notice about a **CD** is that it is shiny on one side and dull on the other.



- **Types of Compact Disk (CD) :**  
These are various types of CD as follows below.
- **CD-R** – It is a blank CD in which data can be stored once which is known as CD-ROM after storing data on it.
- **CD-ROM** – It became a ROM (read only memory), in which you can not update or delete data. Only you can read data using a CD-drive.
- **CD-RW** – You can update or delete data multiple times, if you want to do so.
- **Advantages of CD:**
- It can store data for long time, i.e., durable memory.
- It is a reliable and widely used memory.
- It provides random data access.
- It can not be affected by the magnetic field.
- It is economical, i.e., stores more data in less space.

- **Disadvantages of CD:**

- It is a good memory but slower than a disk.
- Writing or copying data on CD is not easy.
- The surface of a **CD** is made of a polycarbonate layer with molded spiral tracks on the top. The **data** are stored on the **CD** as a series of minute grooves which are known as 'pits' encoded on these spiral tracks. The areas between the 'pits' are known as 'lands'. ... A **CD** burner is used to write (burn) the **data** on a **CD**

**Optical** storage devices save **data** as patterns of dots that can be **read** using light. A laser beam is the usual light source. The **data** on the storage medium is **read** by bouncing the laser beam off the surface of the medium. ... This process is known as 'burning' **data** onto a **disc**.

### Writing Data

•**Optical** storage devices save **data** as patterns of dots that can be **read** using light. A laser beam is the usual light source. The **data** on the storage medium is **read** by bouncing the laser beam off the surface of the medium. ... This process is known as 'burning' **data** onto a **disc**.

### •Reading Data

•**Optical** storage devices save **data** as patterns of dots that can be **read** using light. A laser beam is the usual light source. The **data** on the storage medium is **read** by bouncing the laser beam off the surface of the medium. ... This process is known as 'burning' **data** onto a **disc**.

### Input Devices



- In computing, an **input device** is a piece of equipment used to provide data and control signals to an information processing system such as a computer or information appliance. Examples of **input devices** include keyboards, mouse, scanners, digital cameras, joysticks, and microphones.

## Keyboard



- Keyboard is most commonly used input device
- It converts Human Language in to machine language
- The most commonly used keyboard is the IBM-101 keyboard. The keys are arranged in different groups
- 
- 1) Alphanumeric keys 2) Numeric Key pad 3) Cursor Movement Keys 4) function Keys 5) modifier Keys 6) special Purpose Keys
- Alpha Numeric Keys This set Contains alphabetic ,numeric,and special Characters
- Numeric Key pad :- This set of keys act as numeric keys when numlock is on and act cursor moverment when numlock is off
- Cursor Movement keys :- The four Keys up,down ,left, right for moving cursor
- Function keys :- The 12 keys F1 to F12 represent on peration . The purpose of each key depends on the application being used
- Special purpose keys:-

- These keys perform specialised functions . They are Esc, Pause,Insert,Delete etc
- Modifier keys:- They are used to modify the input of other keys they are used by holding down the modifier key while pressing another . They are Shift ,Ctrl and Alt
- Mouse



•

#### Light Pen



- It is a pen based system we can hold the pen in our hand and directly point with it on the screen to select menu items or icons or directly draw graphics on the screen



Windows XP/ 7/ 8/ 10 OS



- A joystick is a pointing device. To make the movement of the spherical ball easier the spherical ball which moves in a socket has a stick mounted on it. The stick can be moved forward or backward, left or right to move and position the graphics cursor at the desired position. On most joysticks a button on the top is provided to select the option which is currently pointed to by the cursor. The button is clicked to make this selection. Typical uses of joysticks include video games, for controlling industrial robots.

## Scanner



- A Scanner is an input device. Which translates paper documents into an electronic format, which can be stored in a computer.
- The input documents may be typed text, pictures, graphics
- Scanners come in various shapes and sizes.
- The two commonly used types are
- 1) flatbed Scanner

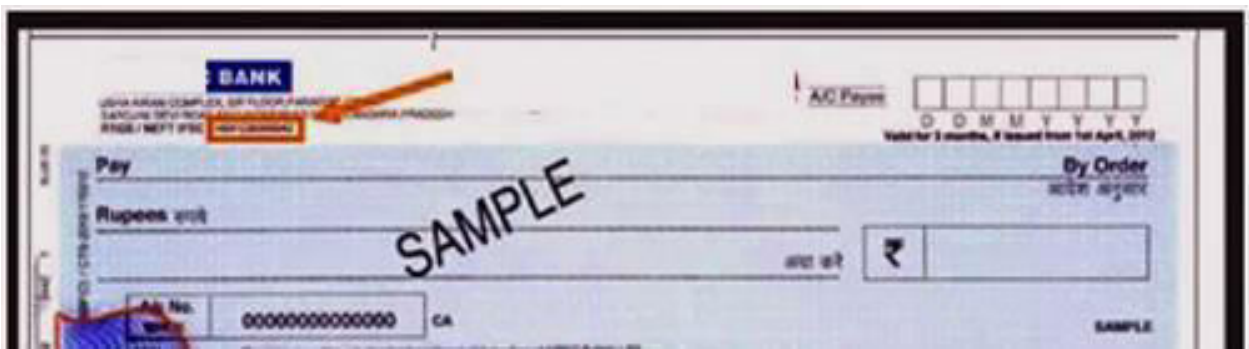


- It is like a copy machine Which consists of a box having a glass plate on its top The document to be scanned is placed upside down on the glass plate The light source is situate below the glass plate and moves horizontally from left to right and right to left
- When activated

Hand held scanner



- It can be held in hand so it is called as hand held scanner
- To scan a document the scanner is slowly dragged from one end of the document to its other end with its light on .







**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh52**

**73**

**@gmail.com**





**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of Management**  
**Career Development and Research.**

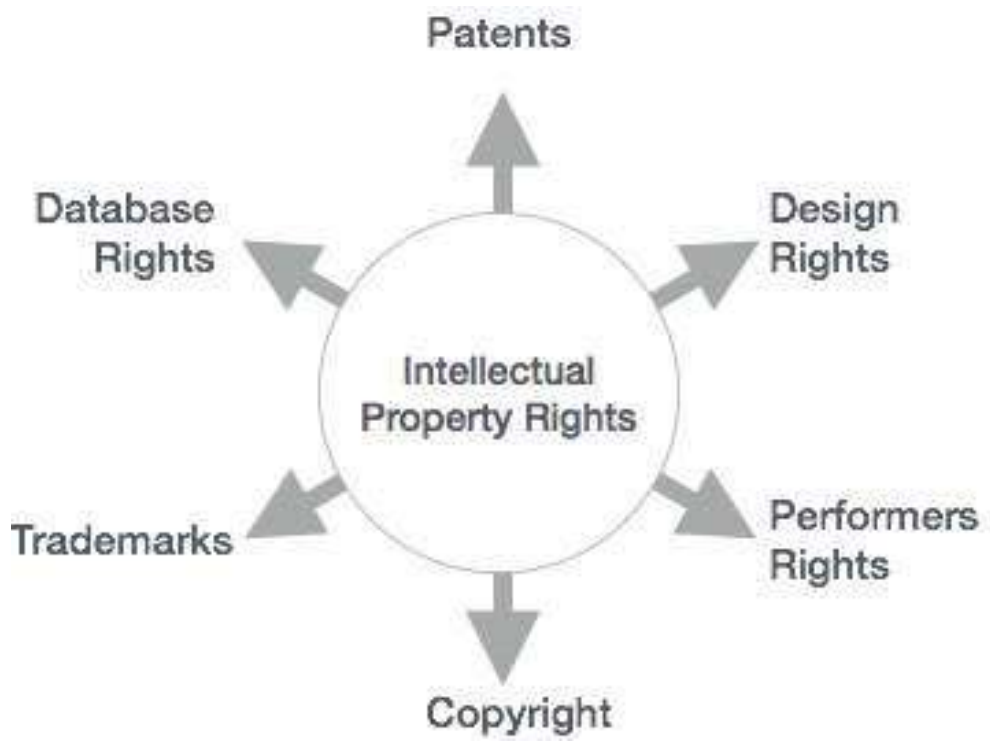
**156-B Railway Lines Solapur.**  
**Affiliated PAH Solapur University**

Name of Faculty . Mr Santosh Kulkarni

Subject Cyber Law Topic No 2  
Programme:- BCA I Sem I

# Security Aspect Aspect

- **Electronic data** and its transmission are vulnerable to unauthorised interference from criminals and persons having vested interests. Ensuring security of data through legal and technical means has become a matter of concern. A legal infrastructure has become imperative to protect data and information.



- Intellectual Property Rights (IPR) and Cyber Laws cannot be separated, and online content must be protected. ... Those rights are protected under IPR. **Patent, Copyright, Trademarks, Trade Secrets, Industrial and Layout Designs, Geographical Indications etc.** are intellectual property rights.

# Criminal Aspect

- Cyber crimes can involve criminal activities that are traditional in nature, such as **theft, fraud, forgery, defamation and mischief**, all of which are subject to the Indian Penal Code. The crime of computers has also given birth to a gamut of new age crimes that are addressed by the Information Technology Act, 2000.

Faculty- Nilima Mondhe

**Class - BCA-I**

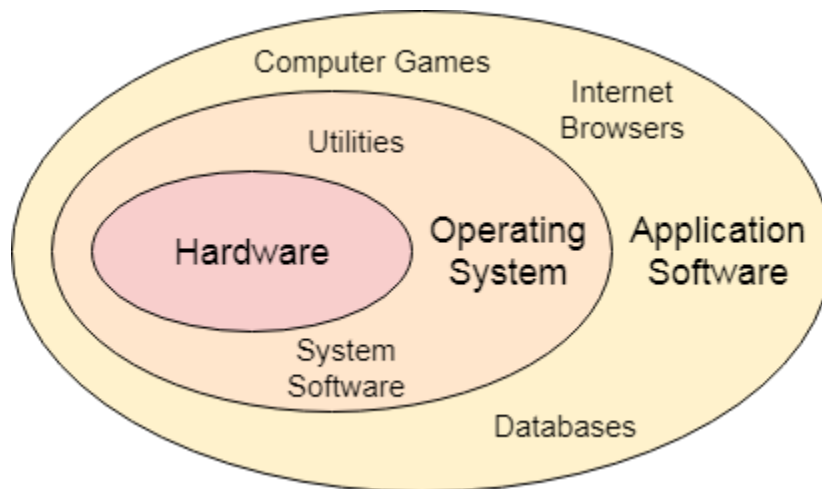
**Sem- II**

**Subject- Operating System**

### Operating System Definition and Function

Computers are made up of of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.

We need a system which can act as an intermediary and manage all the processes and resources present in the system.



An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes,

Resource Allocation,

CPU management,

File Management and many other tasks.

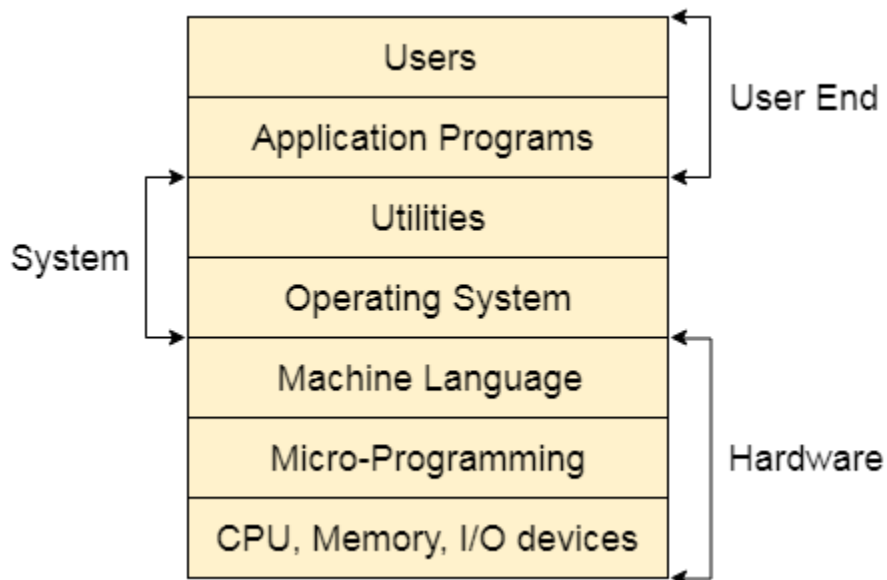
The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

## Structure of a Computer System

A Computer System consists of:

- Users (people who are using the computer)
- Application Programs (Compilers, Databases, Games, Video player, Browsers, etc.)
- System Programs (Shells, Editors, Compilers, etc.)
- Operating System ( A special program which acts as an interface between user and hardware )
- Hardware ( CPU, Disks, Memory, etc)





What does an Operating system do?

1. Process Management
2. Process Synchronization
3. Memory Management
4. CPU Scheduling
5. File Management
6. Security

---

## Types of Operating Systems

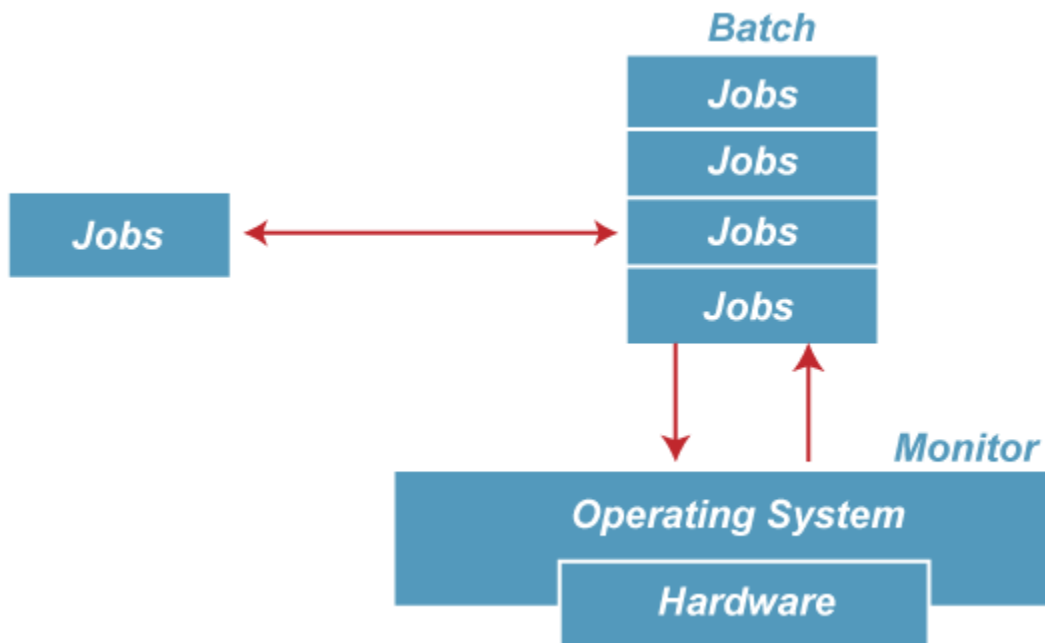
An operating system is a well-organized collection of programs that manages the computer hardware. It is a type of system software that is responsible for the smooth functioning of the computer system.

## Batch Operating System

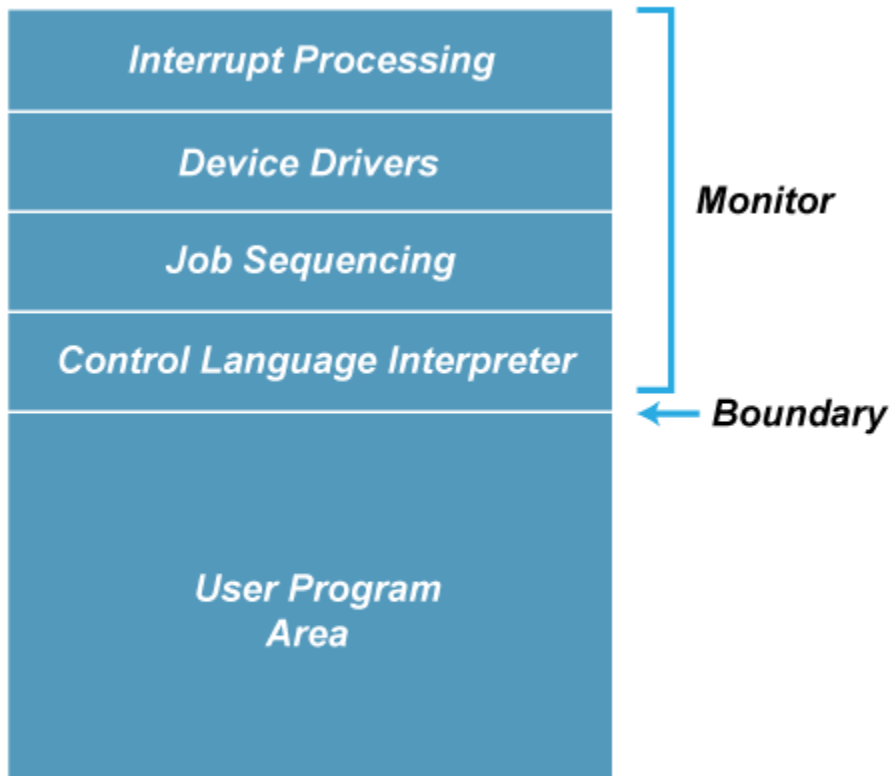
In the 1970s, Batch processing was very popular. In this technique, similar type jobs were batched together and executed in time. People were used to have a single computer which was called a mainframe.

In Batch operating system, access is given to more than one person; they submit their respective jobs to the system for the execution.

The system put all of the jobs in a queue on the basis of first come first served. Then it executes the jobs one by one. The users collect their respective output after all the jobs get executed.



The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory. The remaining part is used for servicing jobs.



**Figure: Memory Layout of the resident monitor**

## Advantages of Batch OS

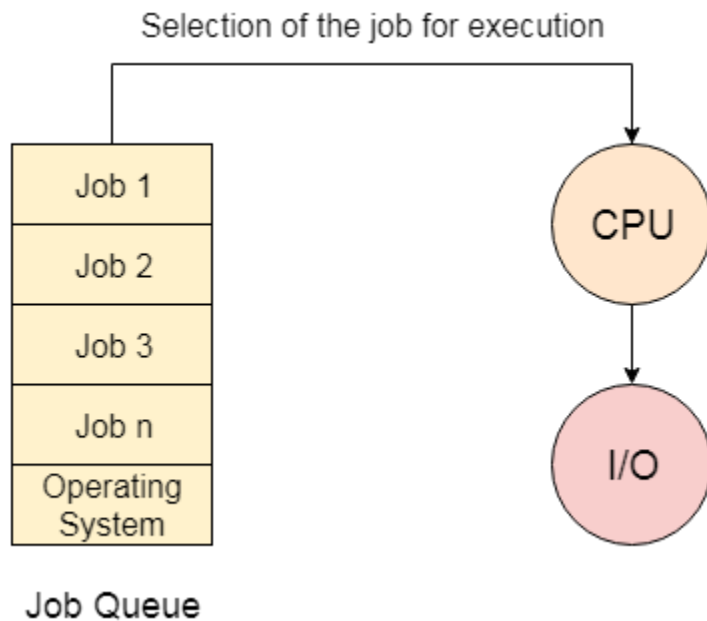
- The use of a resident monitor improves computer efficiency as it eliminates CPU time between two jobs.

## Disadvantages of Batch OS

### 1. Starvation

Batch processing suffers from starvation.

## For Example:



There are five jobs J1, J2, J3, J4, and J5, present in the batch. If the execution of J1 is very high, then the other four jobs will never be executed, or they will have to wait for a very long time. Hence the other processes get starved.

## 2. Not Interactive

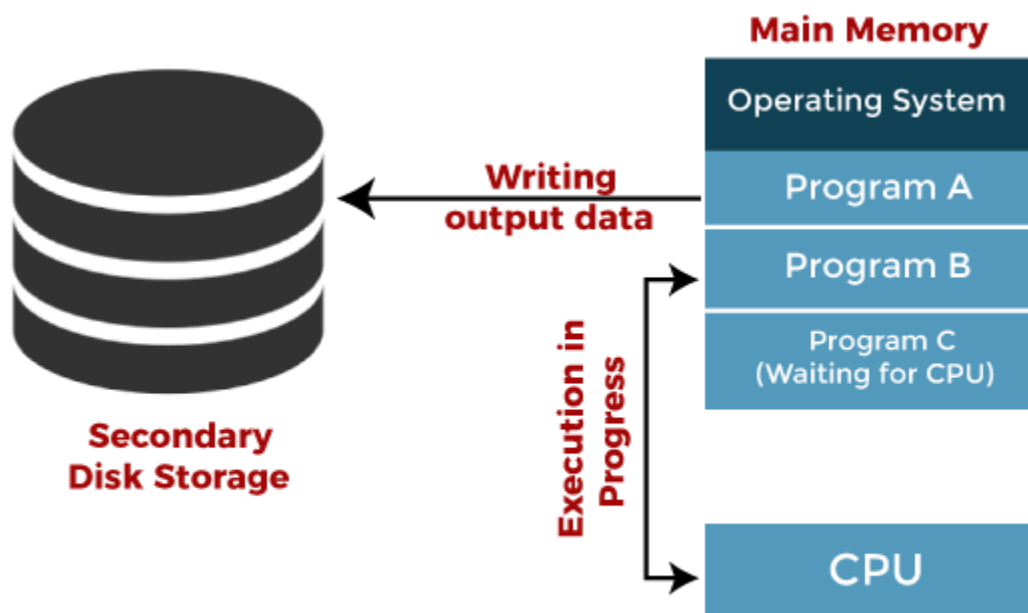
Batch Processing is not suitable for jobs that are dependent on the user's input. If a job requires the input of two numbers from the console, then it will never be executed in the batch processing scenario since the user is not present at the time of execution.

## Multiprogramming Operating System

Multiprogramming is an extension to batch processing where the CPU is a

kept busy. Each process needs two types of system time: CPU time and IO time.

In a multiprogramming environment, when a process does its I/O, the CPU starts the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



*Jobs in multiprogramming system*

### Advantages of Multiprogramming OS

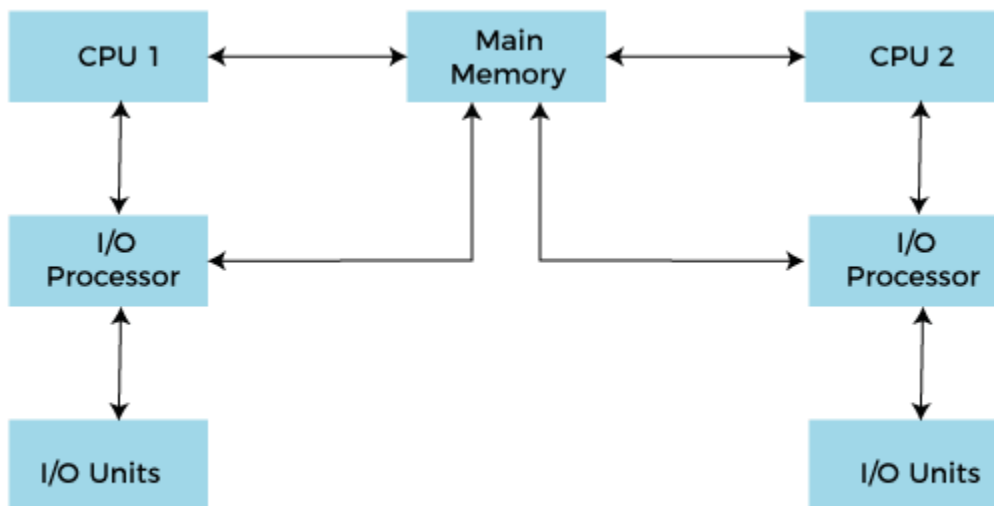
- Throughout the system, it increased as the CPU always had one program to execute.
- Response time can also be reduced.

### Disadvantages of Multiprogramming OS

- Multiprogramming systems provide an environment in which various system resources are used efficiently, but they do not provide any user interaction with the computer system.

### Multiprocessing Operating System

In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.

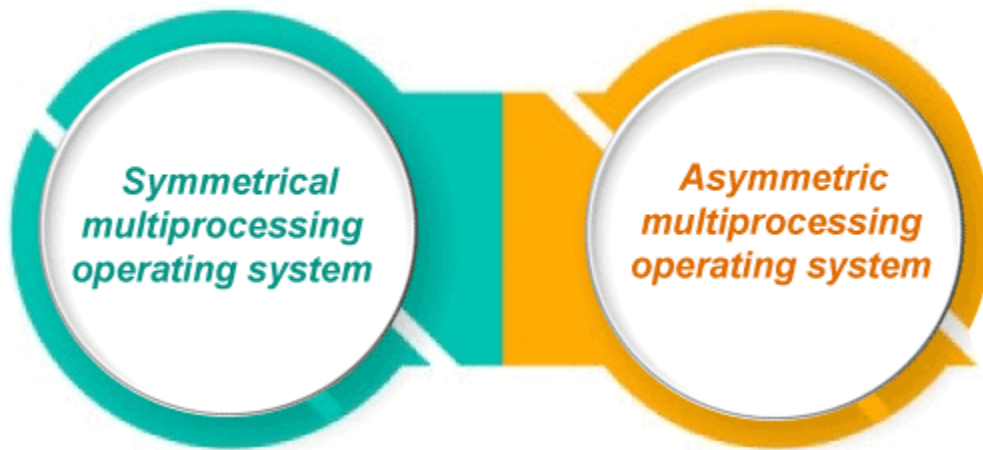


Working of Multiprocessor System

In Multiprocessing, Parallel computing is achieved. More than one processor present in the system can execute more than one process simultaneously, which will increase the throughput of the system.



## **Types of Multiprocessing systems**



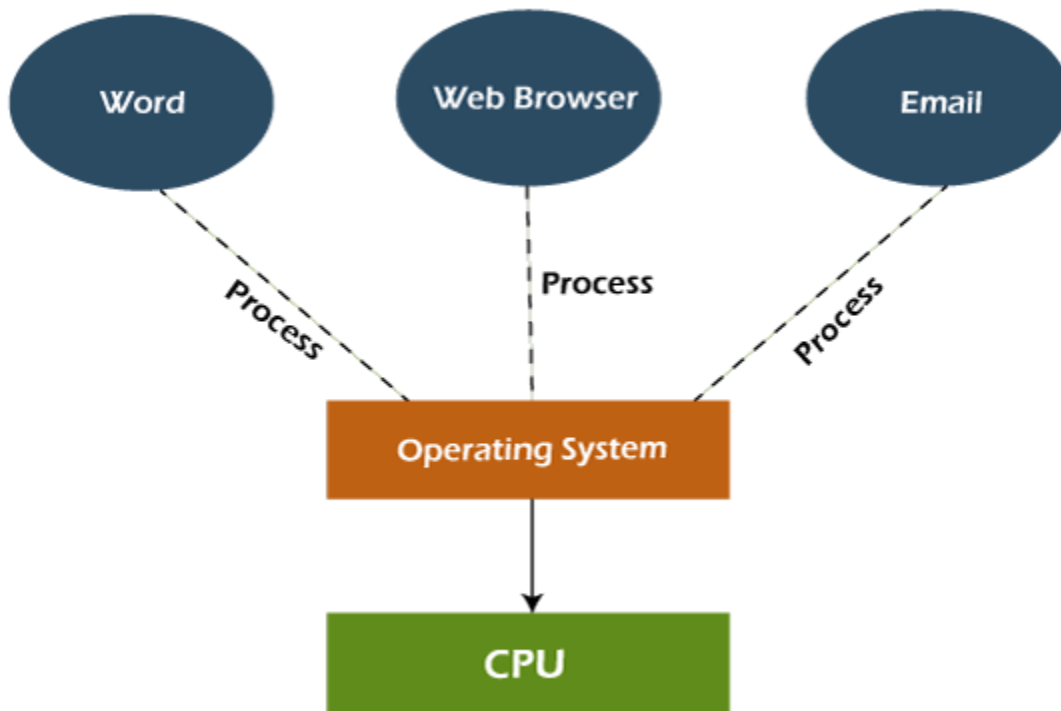
Advantages of Multiprocessing operating system:

- **Increased reliability:** Due to the multiprocessing system, processing can be distributed among several processors. This increases reliability. If one processor fails, the task can be given to another processor for completion.
- **Increased throughput:** As several processors increase, more work can be done in less time.

Disadvantages of Multiprocessing operating System

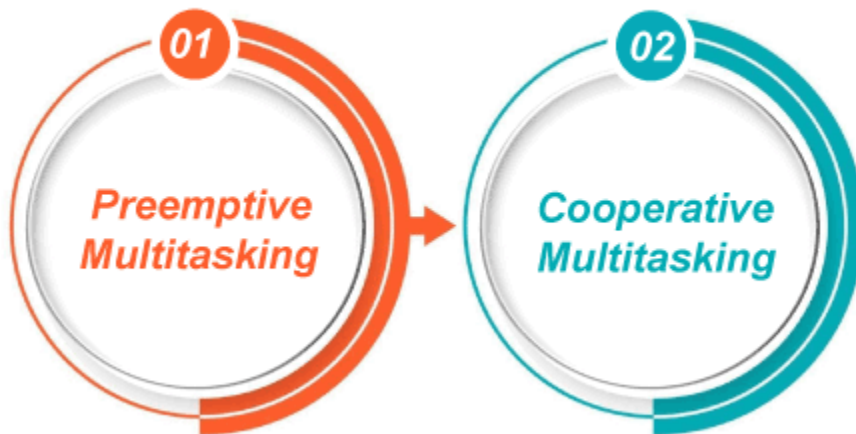
- Multiprocessing operating system is more complex and sophisticated. It takes care of multiple CPUs simultaneously.

**Multitasking Operating System**



The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously. It allows a user to perform more than one computer task at the same time.

## Types of Multitasking



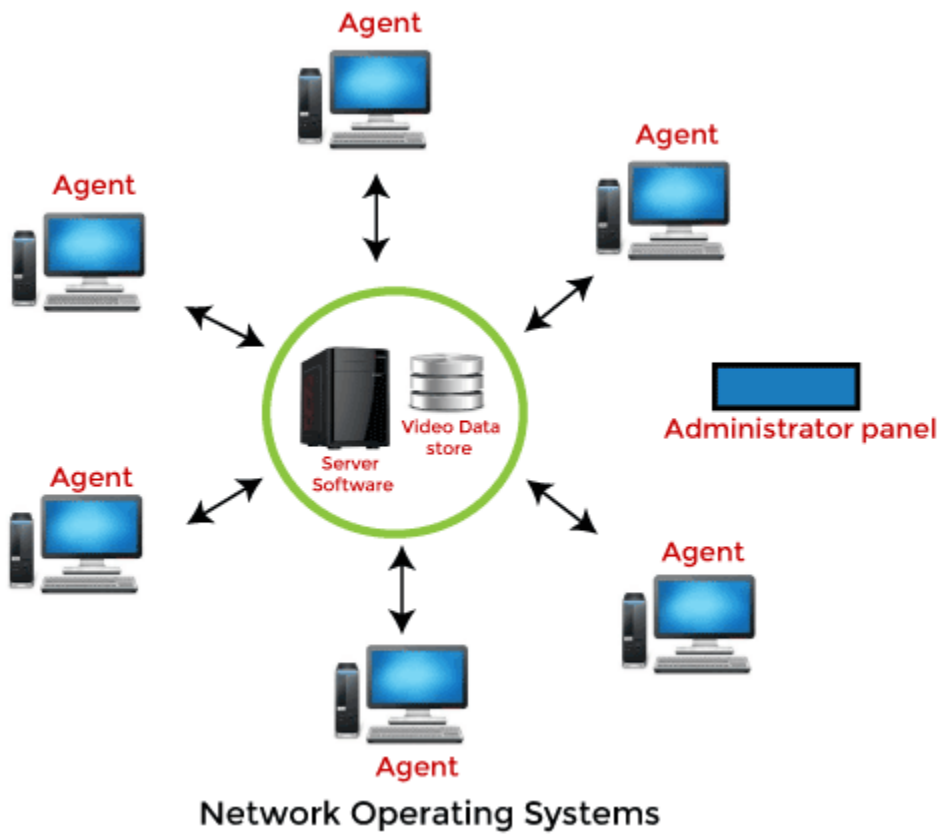
### Advantages of Multitasking operating system

- This operating system is more suited to supporting multiple simultaneously.
- The multitasking operating systems have well-defined memory manage

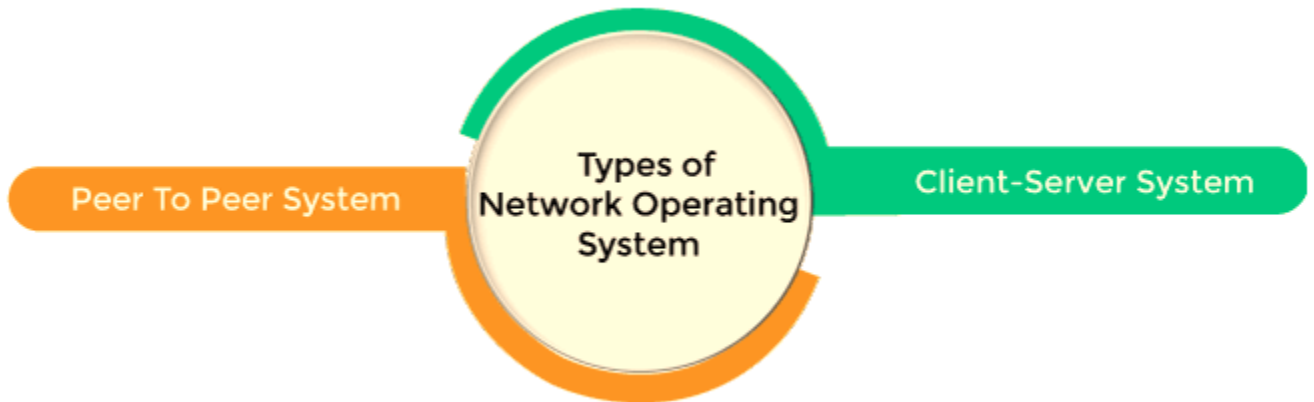
### Disadvantages of Multitasking operating system

- The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

### Network Operating System



An Operating system, which includes software and associated protocols to manage hardware resources and enable computers to communicate with other computers via a network conveniently and effectively, is called Network Operating System.



### Advantages of Network Operating System

- In this type of operating system, network traffic reduces due to the direct communication between clients and the server.
- This type of system is less expensive to set up and maintain.

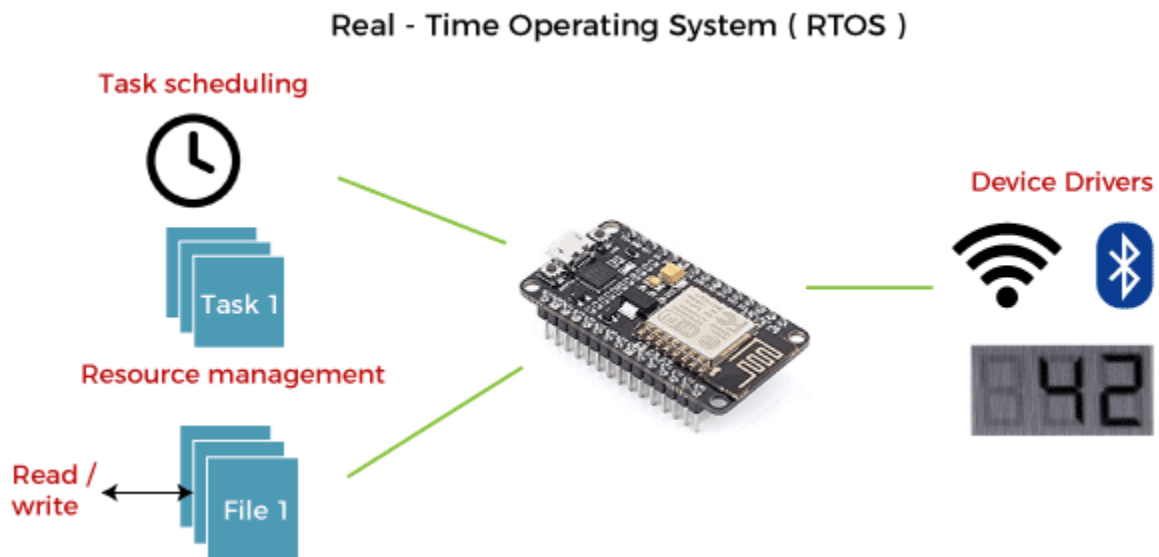
### Disadvantages of Network Operating System

- In this type of operating system, the failure of any node in a system affects the whole system.
- Security and performance are important issues. So trained network administrators are required for network administration.

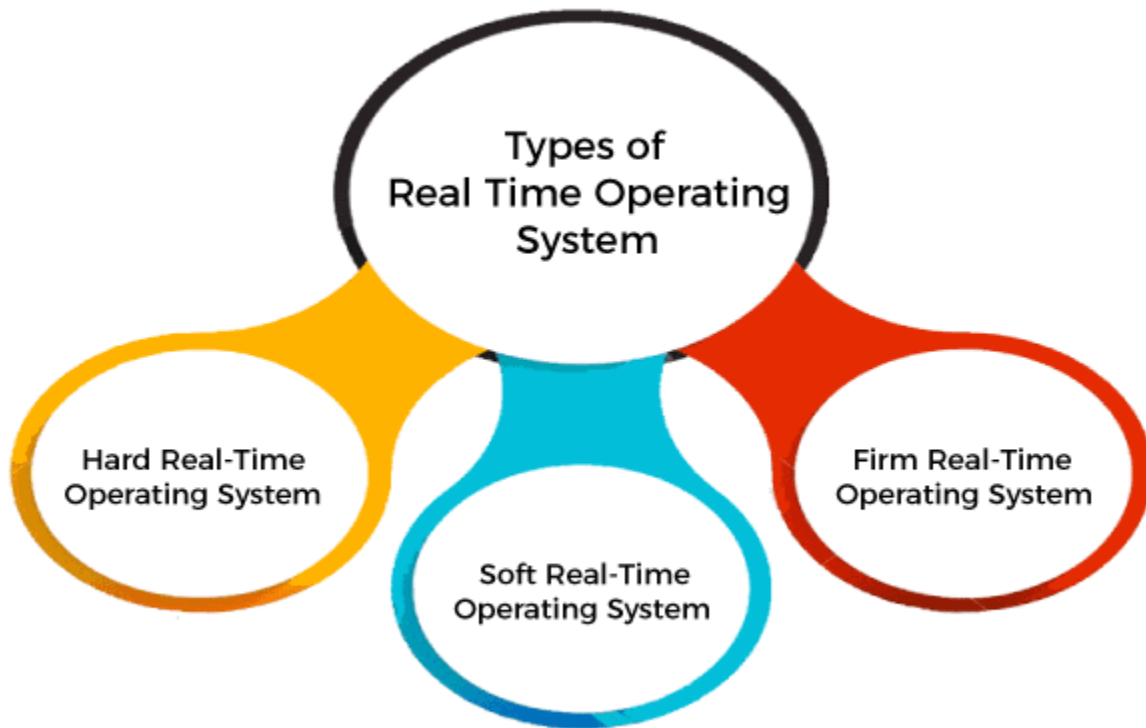
### Real Time Operating System

In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even

result is produced, it will be completely useless.



The Application of a Real-Time system exists in the case of military applications. If you want to drop a missile, then the missile is supposed to be dropped with a certain precision.



### Advantages of Real-time operating system:

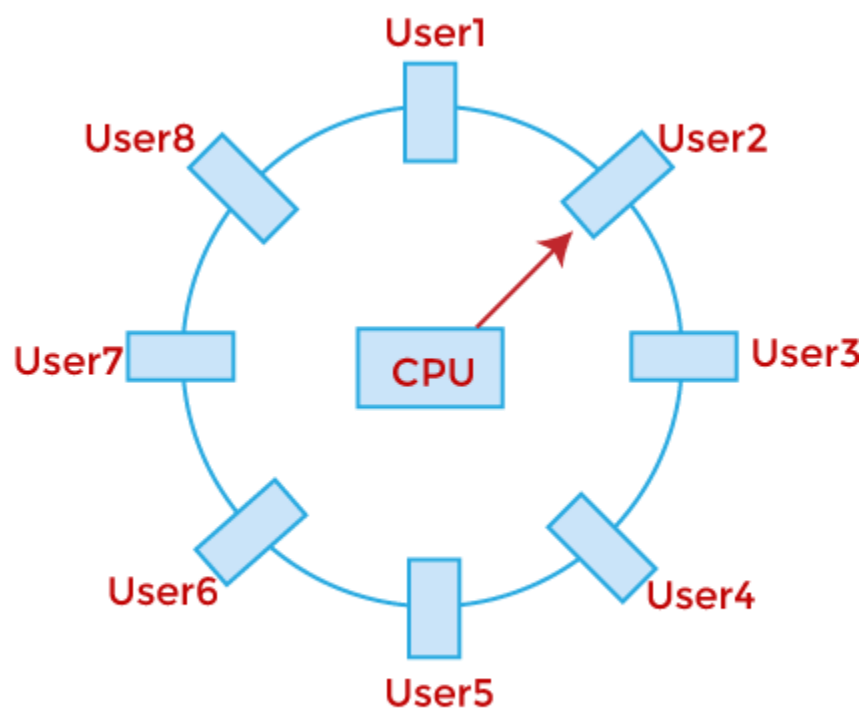
- Easy to layout, develop and execute real-time applications under the time operating system.
- In a Real-time operating system, the maximum utilization of device systems.

### Disadvantages of Real-time operating system:

- Real-time operating systems are very costly to develop.
- Real-time operating systems are very complex and can consume critical cycles.

## Time-Sharing Operating System

In the Time Sharing operating system, computer resources are allocated in a dependent fashion to several programs simultaneously. Thus it helps to provide a large number of user's direct access to the main computer. It is a logical extension of multiprogramming. In time-sharing, the CPU is switched among multiple programs given by different users on a scheduled basis.



**Timesharing in case of 8 users**

A time-sharing operating system allows many users to be served simultaneously, so sophisticated CPU scheduling schemes and Input/output management are required.



Time-sharing operating systems are very difficult and expensive to build.

### Advantages of Time Sharing Operating System

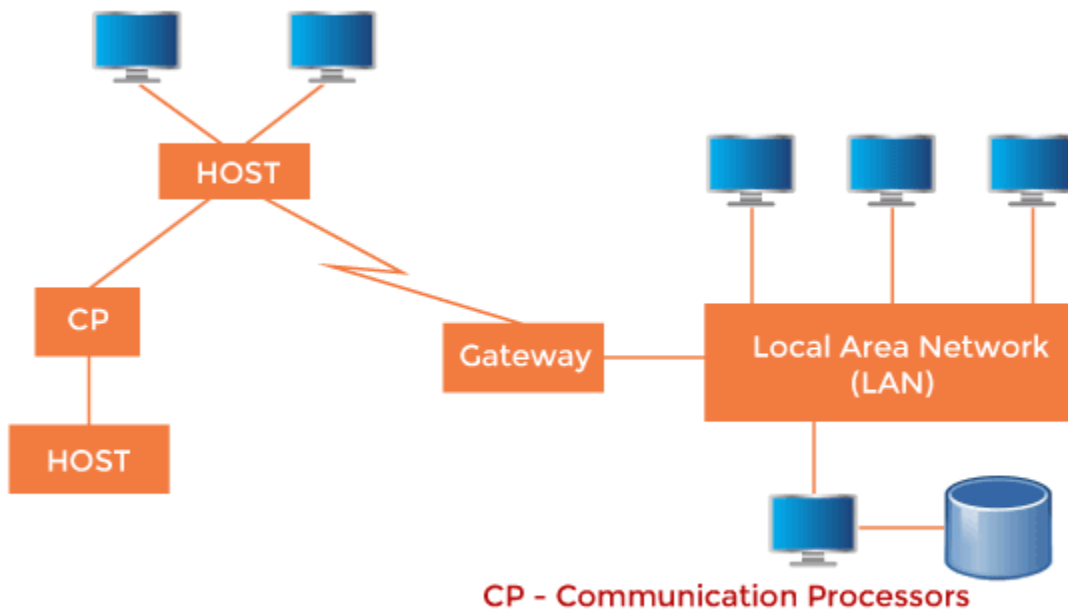
- The time-sharing operating system provides effective utilization and sharing of resources.
- This system reduces CPU idle and response time.

### Disadvantages of Time Sharing Operating System

- Data transmission rates are very high in comparison to other methods.
- Security and integrity of user programs loaded in memory and data need to be maintained as many users access the system at the same time.

### Distributed Operating System

The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines. A part of the distributed Operating system is installed on each machine to make communication possible. Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.



**A Typical View of a Distributed System**

## Advantages of Distributed Operating System

- The distributed operating system provides sharing of resources.
- This type of system is fault-tolerant.

## Disadvantages of Distributed Operating System

- Protocol overhead can do

## Process Management in OS

A Program does nothing unless its instructions are executed by a CPU. A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

The operating system is responsible for the following activities in connection with Process Management

1. Scheduling processes and threads on the CPUs.
2. Creating and deleting both user and system processes.
3. Suspending and resuming processes.
4. Providing mechanisms for process synchronization.
5. Providing mechanisms for process communication.

### Attributes of a process

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.

#### 1. Process ID

When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

## 2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

## 3. Process State

The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. We will discuss about them later in detail.

## 4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

## 5. General Purpose Registers

Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

## 6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

### 7. List of open devices

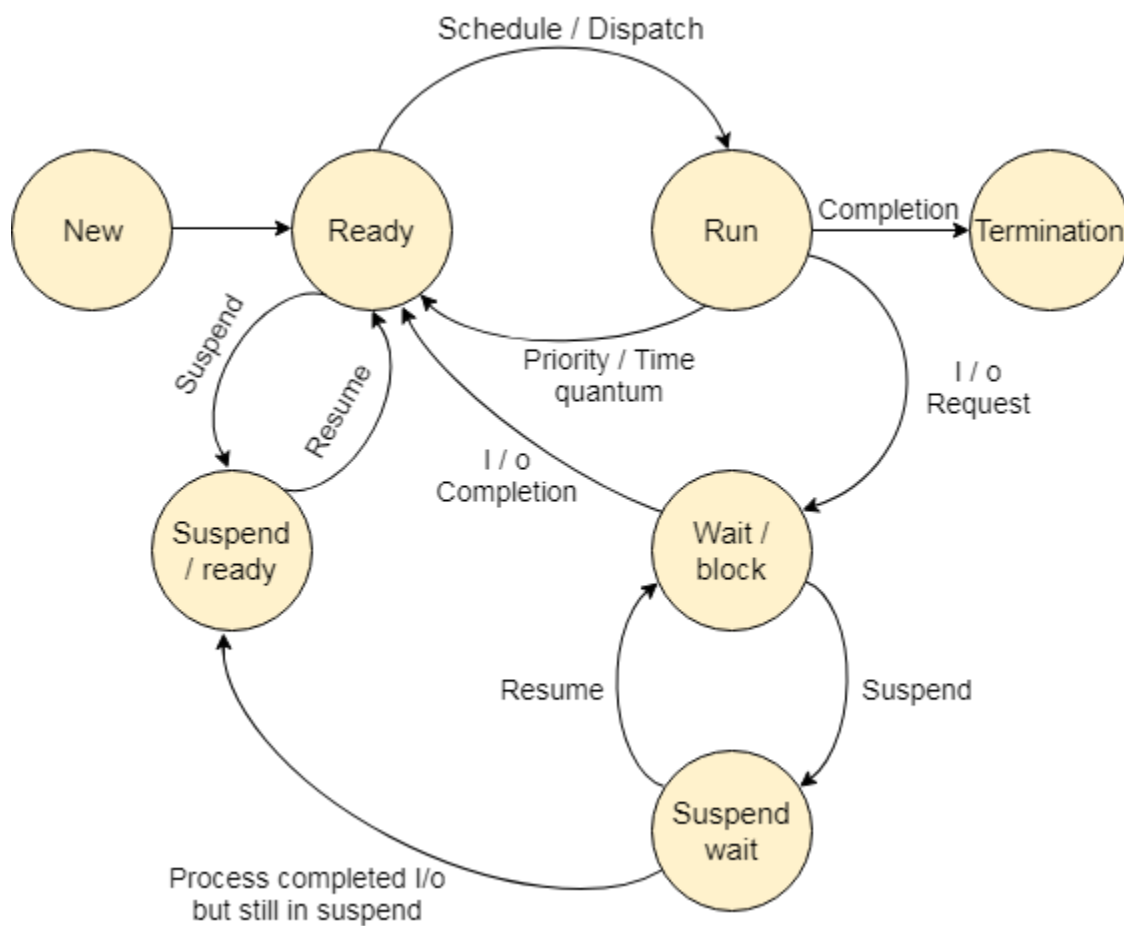
OS also maintain the list of all open devices which are used during the execution of the process.

<b>Process ID</b>
<b>Program Counter</b>
<b>Process State</b>
<b>Priority</b>
<b>General Purpose Registers</b>
<b>List of Open Files</b>
<b>List of Open Devices</b>

**Process Attributes**

## Process States

### State Diagram



The process, from its creation to completion, passes through various states. The minimum number of states is five.

The names of the states are not standardized although the process may be in one of the following states during execution.

### 1. New

A program which is going to be picked up by the OS into the main memory is called a new process.

### 2. Ready

Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory.

The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.

### 3. Running

One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for

a particular time will always be one. If we have  $n$  processors in the system then we can have  $n$  processes running simultaneously.

#### 4. Block or wait

From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.

When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.

#### 5. Completion or termination

When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

#### 6. Suspend ready

A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out



into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available.

## 7. Suspend wait

Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for the higher priority process. These processes complete their execution once the main memory gets available and their wait is finished.

## Operations on the Process

### 1. Creation

Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for the execution.

### 2. Scheduling

Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it. Selecting the process which is to be executed next, is known as scheduling.

### 3. Execution

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

### 4. Deletion/killing

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

### Scheduling Algorithms

There are various algorithms which are used by the Operating System to schedule the processes on the processor in an efficient way.

### The Purpose of a Scheduling algorithm

1. Maximum CPU utilization
2. Fair allocation of CPU
3. Maximum throughput
4. Minimum turnaround time
5. Minimum waiting time
6. Minimum response time

There are the following algorithms which can be used to schedule the jobs.

### 1. First Come First Serve

It is the simplest algorithm to implement. The process with the minimal arrival time will get the CPU first. The lesser the arrival time, the sooner will the process gets the CPU. It is the non-preemptive type of scheduling.

### 2. Round Robin

In the Round Robin scheduling algorithm, the OS defines a time quantum (slice). All the processes will get executed in the cyclic way. Each of the process will get the CPU for a small amount of time (called time quantum) and then get back to the ready queue to wait for its next turn. It is a preemptive type of scheduling.

### 3. Shortest Job First

The job with the shortest burst time will get the CPU first. The lesser the burst time, the sooner will the process get the CPU. It is the non-preemptive type of scheduling.

### 4. Shortest remaining time first

It is the preemptive form of SJF. In this algorithm, the OS schedules the Job according to the remaining time of the execution.

## 5. Priority based scheduling

In this algorithm, the priority will be assigned to each of the processes. The higher the priority, the sooner will the process get the CPU. If the priority of the two processes is same then they will be scheduled according to their arrival time.

## 6. Highest Response Ratio Next

In this scheduling Algorithm, the process with highest response ratio will be scheduled next. This reduces the starvation in the system.

## FCFS Scheduling

**First come first serve** (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

## Advantages of FCFS

- Simple
- Easy
- First come, First serv

## Disadvantages of FCFS

1. The scheduling method is non preemptive, the process will run to the completion.
2. Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
3. Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

### Example

Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID **P0, P1, P2, P3 and P4**. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

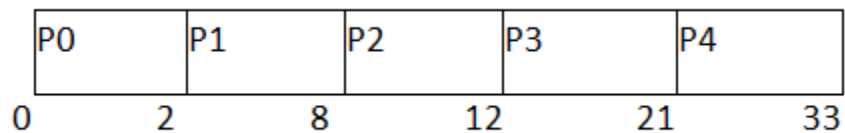
The Turnaround time and the waiting time are calculated by using the following formula.

1. Turn Around **Time** = **Completion** Time - Arrival Time
2. Waiting **Time** = **Turnaround** time - Burst Time

The average waiting Time is determined by summing the respective waiting time of all the processes and divided the sum by the total number of processes.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Avg Waiting Time =  $31/5$



**(Gantt chart)**

In the Example, We have 3 processes named as **P1, P2 and P3**. The Burt Time of process P1 is highest.

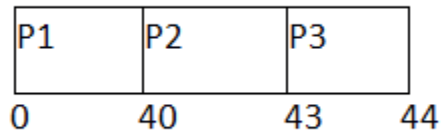
The Turnaround time and the waiting time in the following table, are calculated by the formula,

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

In the First scenario, The Process P1 arrives at the first in the queue although; the burst time of the process is the highest among all. Since, the Scheduling algorithm, we are following is FCFS hence the CPU will execute the Process P1 first.

In this schedule, the average waiting time of the system will be very high. That is because of the convoy effect. The other processes P2, P3 have to wait for their turn for 40 units of time although their burst time is very low. This schedule suffers from starvation.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	40	40	40	0
2	1	3	43	42	39
3	1	1	44	43	42

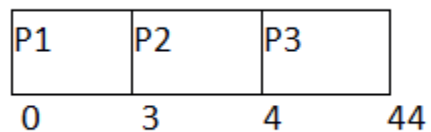


Avg waiting Time =  $81/3$

In the Second scenario, If Process P1 would have arrived at the last of the queue and the other processes P2 and P3 at earlier then the problem of starvation would not be there.

Following example shows the deviation in the waiting times of both the scenarios. Although the length of the schedule is same that is 44 units but the waiting time will be lesser in this schedule.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	40	44	43	3
2	0	3	3	3	0
3	0	1	4	4	3





## **Avg Waiting Time=6/3**

### **Shortest Job First (SJF) Scheduling**

Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

### **Advantages of SJF**

1. Maximum throughput
2. Minimum average waiting and turnaround time

### **Disadvantages of SJF**

1. May suffer with the problem of starvation
2. It is not implementable because the exact Burst time for a process can't be known in advance.

There are different techniques available by which, the CPU burst time of the process can be determined. We will discuss them later in detail.

### Example

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Since, No Process arrives at time 0 hence; there will be an empty slot in the **Gantt chart** from time 0 to 1 (the time at which the first process arrives).

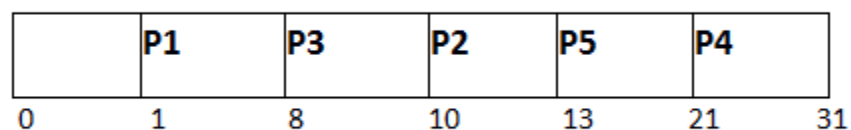
According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.

Till now, we have only one process in the ready queue hence the scheduler will schedule this to the processor no matter what is its burst time.

This will be executed till 8 units of time. Till then we have three more processes arrived in the ready queue hence the scheduler will choose the process with the lowest burst time.

Among the processes given in the table, P3 will be executed next since it is having the lowest burst time among all the available processes.

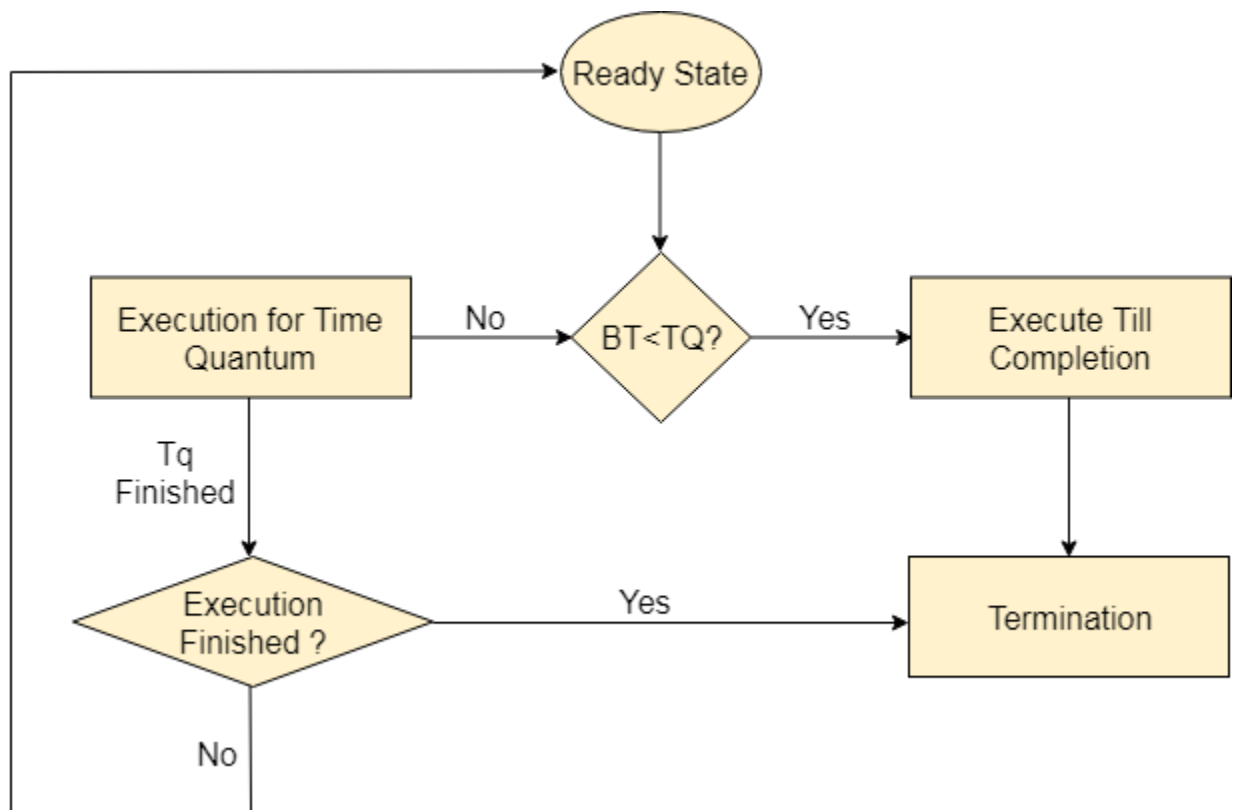
So that's how the procedure will go on in **shortest job first (SJF)** scheduling algorithm.



Avg Waiting Time =  $27/5$

## Round Robin Scheduling Algorithm

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the **preemptive version** of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**. A certain time slice is defined in the system which is called time **quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.



## Advantages

1. It can be actually implementable in the system because it is not depending on the burst time.
2. It doesn't suffer from the problem of starvation or convoy effect.
3. All the jobs get a fare allocation of CPU.

## Disadvantages

1. The higher the time quantum, the higher the response time in the system.
2. The lower the time quantum, the higher the context switching overhead in the system.
3. Deciding a perfect time quantum is really a very difficult task in the system.

## Introduction

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.



**Prin. K. P. Mangalvedhekar Institute of Management  
Career Development and Research.  
156-B Railway Lines Solapur.**

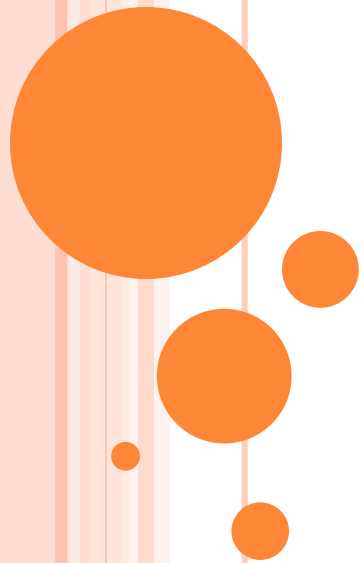
**BCA –I – Sem. –II Subject-Web Technology  
Chapter 2**

**By : Mrs Nilima Mondhe**



# JQUERY

**Created by Nilima Mondhe  
BCA Dept**



# OBJECTIVE

To study basic concepts of jquery and jquery programming.



# INTRODUCTION TO JQUERY

- jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: **Write less, do more**. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery –
- **DOM manipulation** – The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
- **Event handling** – The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- **AJAX Support** – The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.
- **Animations** – The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- **Lightweight** – The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- **Cross Browser Support** – The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology** – The jQuery supports CSS3 selectors and basic XPath syntax.



# NEED OF JQUERY

There is a need of jquery because,

- It is very fast and extensible.
- It facilitates the users to write UI related function codes in minimum possible lines.
- It improves the performance of an application.
- Browser's compatible web applications can be developed.
- It uses mostly new features of new browsers.
- So, we can say that out of the lot of JavaScript frameworks, jQuery is the most popular and the most extendable. Many of the biggest companies on the web use jQuery-
- Google
- Microsoft
- IBM
- Netflix



# ADDING JQUERY TO WEB PAGES

- There are two ways to use jQuery.
- **Local Installation** – You can download jQuery library on your local machine and include it in your HTML code.
- **CDN Based Version** – You can include(URL) jQuery library into your HTML code directly from Content Delivery Network (CDN).



# LOCAL INSTALLATION

```
<html>
  <head>
<title>The jQuery Example</title>
  <script type = "text/javascript" src = "/jquery/jquery-
    2.1.3.min.js">
  </script>
<script type = "text/javascript">
$(document).ready(
function() { document.write("Hello, World!"); }
);
  </script> </head> <body> <h1>Hello</h1>
</body> </html>
```



# CDN BASED VERSION

```
<html> <head>
<title>The jQuery Example</title>
<script type = "text/javascript" src =
  "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/
  jquery.min.js"> </script>
<script type = "text/javascript">
  $(document).ready(function() {
    document.write("Hello, World!"); });
</script> </head>
<body> <h1>Hello</h1> </body>
</html>
```



# JQUERY SYNTAX

- With jQuery we select (query) HTML elements and perform "actions" on them.
- Basic syntax is: **`$(selector).action()`**
  1. A \$ sign to define/access jQuery
  2. A (*selector*) to "query (or find)" HTML elements
  3. A jQuery *action()* to be performed on the element(s)
- Examples:
  1. `$(this).hide()` - hides the current element.
  2. `$("p").hide()` - hides all `<p>` elements.
  3. `$(".test").hide()` - hides all elements with `class="test"`.
  4. `$("#test").hide()` - hides the element with `id="test"`.





# The Document Ready Event

jQuery methods syntax

```
$(document).ready(function){
```

```
    // jQuery methods go here...
```

```
});
```

Or

```
$(function){
```

```
    // jQuery methods go here...
```

```
});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).



# JQUERY SELECTORS

- jQuery selectors are one of the most important parts of the jQuery library.
- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing [CSS Selectors](#), and in addition, it has some own custom selectors.
- All selectors in jQuery start with the dollar sign and parentheses: `$()`.



## The element Selector

- The jQuery element selector selects elements based on the element name.
- Eg - `$("p")`

## The #id Selector

- The jQuery *#id* selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the *#id* selector when you want to find a single, unique element.
- Eg- `$("#test")`

## The .class Selector

- The jQuery *.class* selector finds elements with a specific class.
- To find elements with a specific class, write a period character, followed by the name of the class:
- Eg `$(".test")`



## More examples of selectors

`$("*")`---Selects all elements

`$(this)`---Selects the current HTML

`$("#p.intro")`---Selects all `<p>` elements with `class="intro"`

### For eg

```
<html>
```

```
<head>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>
```

```
$(document).ready(function(){
```

```
  $("#button").click(function(){
```

```
    $("#p").hide();
```

```
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<h2>This is a heading</h2>
```

```
<p>This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

```
<button>Click me to hide paragraphs</button>
```

```
</body>
```

```
</html>
```



# JQUERY EVENT METHODS

- The action made on web page which gives response is known as events
- An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element



# HERE ARE SOME COMMON DOM EVENTS:

<b>Mouse Events</b>	<b>Keyboard Events</b>	<b>Form Events</b>	<b>Document/Window Events</b>
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload



# JQUERY SYNTAX FOR EVENT METHODS

- In jQuery, most DOM events have an equivalent jQuery method.
- 1. Click event-**To assign a click event to all paragraphs on a page, we can do this:
  - `$("p").click();`
  - The next step is to define what should happen when the event fires. You must pass a function to the event:
  - `$("p").click(function(){  
 // action goes here!!  
});`



- Click event eg

```
<html>
```

```
<head>
```

```
<script
```

```
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
```

```
</script>
```

```
<script>
```

```
$(document).ready(function(){
```

```
  $("p").click(function(){
```

```
    $(this).hide();
```

```
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>If you click on me, I will disappear.</p>
```

```
<p>Click me away!</p>
```

```
<p>Click me too!</p>
```

```
</body>
```

```
</html>
```





## 2. `$(document).ready()`

- The `$(document).ready()` method allows us to execute a function when the document is fully loaded.

## 3. `dblclick()`

- The `dblclick()` method attaches an event handler function to an HTML element.
- The function is executed when the user double-clicks on the HTML element:

Eg

```
$("p").dblclick(function(){  
    $(this).hide();  
});
```



#### 4. **mouseenter()**

- The `mouseenter()` method attaches an event handler function to an HTML element.
- The function is executed when the mouse pointer enters the HTML element:

Eg-<html>

```
<head>
```

```
<script  
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">  
</script>
```

```
<script>
```

```
$(document).ready(function(){  
  $("#p1").mouseenter(function(){  
    alert("You entered p1!");  
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p id="p1">Enter this paragraph.</p>
```

```
</body>
```

```
</html>
```



## 5. **mouseleave()**

- The `mouseleave()` method attaches an event handler function to an HTML element.
- The function is executed when the mouse pointer leaves the HTML element:

Eg-

```
$("#p1").mouseleave(function(){  
    alert("Bye! You now leave p1!");  
});
```

6. **The mousedown()** method attaches an event handler function to an HTML element.

- The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Eg `$("#p1").mousedown(function){`  
 `alert("Mouse down over p1!");`  
`});`



## 7. **mouseup()**

- The `mouseup()` method attaches an event handler function to an HTML element.
- The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:
- Eg-

```
$("#p1").mouseup(function(){  
    alert("Mouse up over p1!");  
});
```

## 8. **hover()**

- The `hover()` method takes two functions and is a combination of the `mouseenter()` and `mouseleave()` methods.
- Eg-

```
$("#p1").hover(function(){  
    alert("You entered p1!");  
},  
function(){  
    alert("Bye! You now leave p1!");  
});
```

## 9. **focus()**

- The `focus()` method attaches an event handler function to an HTML form field.
- The function is executed when the form field gets focus:



Eg

```
<script>
$(document).ready(function(){
  $("input").focus(function(){
    $(this).css("background-color", "yellow");
  });
  $("input").blur(function(){
    $(this).css("background-color", "green");
  });
});
</script>
</head>
<body>
```

Name: <input type="text" name="fullname"><br>

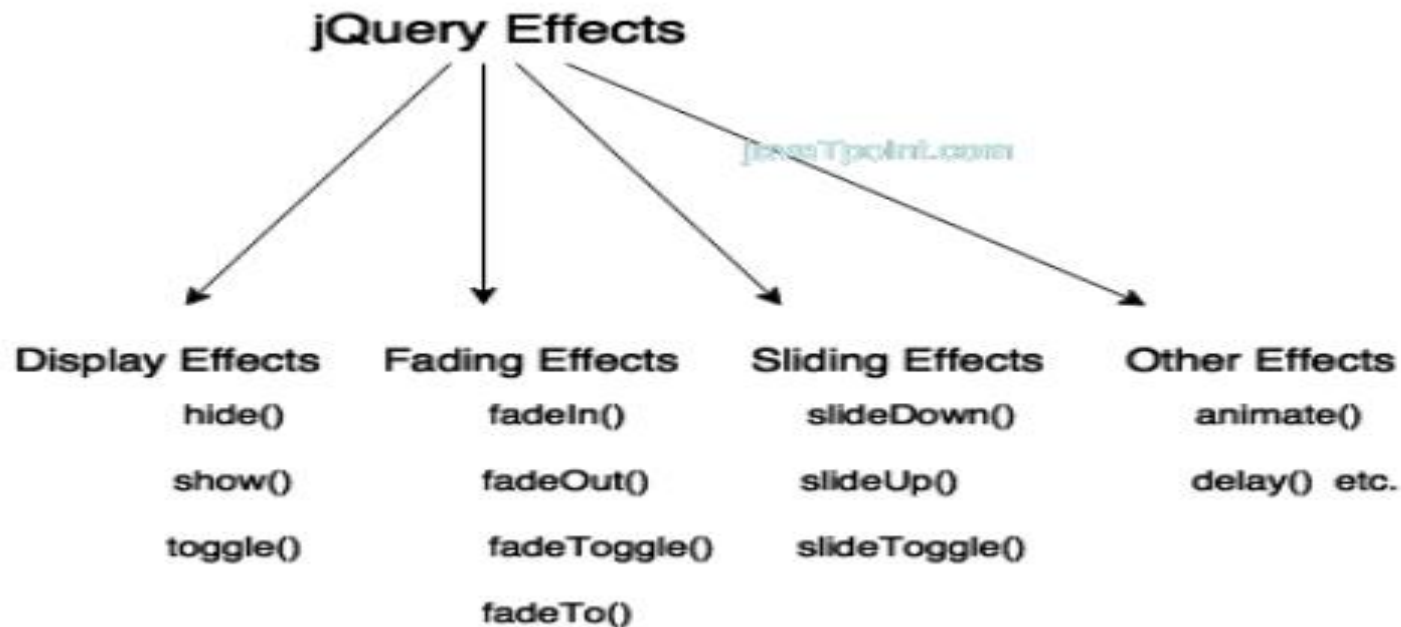
Email: <input type="text" name="email">

```
</body>
```



# JQUERY EFFECTS-HIDE AND SHOW,FADING,SLIDING,ANIMATION

JQUERY ENABLES US TO ADD EFFECTS ON A WEB PAGE. JQUERY EFFECTS CAN BE CATEGORIZED INTO FADING, SLIDING, HIDING/SHOWING AND ANIMATION EFFECTS.



# JQUERY HIDE AND SHOW

- The jQuery `hide()` method is used to hide the selected elements.
- Syntax

```
$(selector).hide(speed,callback);
```

```
$(selector).show(speed,callback);
```

1. The optional `speed` parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
2. **callback**: It is also an optional parameter. It specifies the function to be called after completion of `hide()` or `show()` effect.



Eg

```
<script  
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.m  
  in.js"></script>
```

```
<script>
```

```
$(document).ready(function(){
```

```
  $("#hide").click(function(){
```

```
    $("p").hide();
```

```
  });
```

```
  $("#show").click(function(){
```

```
    $("p").show();
```

```
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>If you click on the "Hide" button, I will disappear.</p>
```

```
<button id="hide">Hide</button>
```

```
<button id="show">Show</button>
```

```
</body>
```





# JQUERY EFFECTS - FADING

- With jQuery you can fade an element in and out of visibility.
- jQuery has the following fade methods:
  1. fadeIn()
  2. fadeOut()
  3. fadeToggle()
  4. fadeTo()
- 1. jQuery fadeIn() Method
  - The jQuery fadeIn() method is used to fade in a hidden element.
  - **Syntax:**
  - `$(selector).fadeIn(speed,callback);`



Eg

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
  });
});
</script>
</head>
<body>
<p>Demonstrate fadeIn() with different parameters.</p>
<button>Click to fade in boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
</body>
```



### 3. jQuery fadeToggle() Method

- The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.
- If the elements are faded out, fadeToggle() will fade them in.
- If the elements are faded in, fadeToggle() will fade them out.

#### **Syntax:**

- `$(selector).fadeToggle(speed,callback);`



```
Eg-<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
  </script>
<script>$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeToggle();
    $("#div2").fadeToggle("slow");
    $("#div3").fadeToggle(3000);
  });
});
</script>
<body>
<p>Demonstrate fadeToggle() with different speed parameters.</p>
<button>Click to fade in/out boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-
  color:red;"></div>
<br>
<div id="div2" style="width:80px;height:80px;background-
  color:green;"></div>
<br>
<div id="div3" style="width:80px;height:80px;background-
  color:blue;"></div>
</body>
```



## 4. jQuery fadeTo() Method

- The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).

### **Syntax:**

- `$(selector).fadeTo(speed,opacity,callback);`

The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
  </script>
```

```
<script>
```

```
$(document).ready(function(){
  $("#button").click(function(){
    $("#div1").fadeOut("slow", 0.15);
    $("#div2").fadeOut("slow", 0.4);
    $("#div3").fadeOut("slow", 0.7);
  });
});
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>Demonstrate fadeTo() with different parameters.</p>
```

```
<button>Click to fade boxes</button><br><br>
```

```
<div id="div1" style="width:80px;height:80px;background-
  color:red;"></div><br>
```

```
<div id="div2" style="width:80px;height:80px;background-
  color:green;"></div><br>
```

```
<div id="div3" style="width:80px;height:80px;background-
  color:blue;"></div>
```

```
</body>
```



## 2. jQuery fadeOut() Method

- The jQuery fadeOut() method is used to fade out a visible element.

### Syntax:

- `$(selector).fadeOut(speed,callback);`

```
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
  });
});
</script>
</head>
<body>
<p>Demonstrate fadeOut() with different parameters.</p>
<button>Click to fade out boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;background-
  color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
```



# JQUERY SLIDING

- With jQuery you can create a sliding effect on elements.
- jQuery has the following slide methods:
  1. `slideDown()`
  2. `slideUp()`
  3. `slideToggle()`

## 1. jQuery `slideDown()` Method

- The jQuery `slideDown()` method is used to slide down an element.

### Syntax:

- `$(selector).slideDown(speed,callback);`





```
Eg -<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideDown("slow");
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eccc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
  display: none;
}
</style>
<body>
  <div id="flip">Click to slide down panel</div>
  <div id="panel">Hello world!</div>
</body>
```



## 2. jQuery slideUp() Method

- The jQuery slideUp() method is used to slide up an element.

### **Syntax:**

- `$(selector).slideUp(speed,callback);`



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideUp("slow");
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eccc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
}
</style>
</head>
<body>
<div id="flip">Click to slide up panel</div>
<div id="panel">Hello world!</div>
</body>
```



### 3. jQuery slideToggle() Method

- The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.
- If the elements have been slid down, slideToggle() will slide them up.
- If the elements have been slid up, slideToggle() will slide them down.

#### **syntax**

- `$(selector).slideToggle(speed,callback);`



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideToggle("slow");
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eccc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
  display: none;
}
</style>
<body>
  <div id="flip">Click to slide the panel down or up</div>
  <div id="panel">Hello world!</div>
</body>
```



# JQUERY ANIMATIONS - THE ANIMATE() METHOD

- The jQuery `animate()` method is used to create custom animations.
- **Syntax:**
- `$(selector).animate({params},speed,callback);`
- The required `params` parameter defines the CSS properties to be animated.
- The optional `speed` parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional `callback` parameter is a function to be executed after the animation completes.



```
Eg-<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquer
  y.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({left: '250px'});
  });
});
</script>
</head>
<body>
<button>Start Animation</button>
<p>By default, all HTML elements have a static position, and
cannot be moved. To manipulate the position, remember to
first set the CSS position property of the element to
relative, fixed, or absolute!</p>
<div
  style="background:#98bf21;height:100px;width:100px;posit
  ion:absolute;"></div>
</body>
```



# JQUERY CALLBACK FUNCTIONS

- JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
- To prevent this, you can create a callback function.
- A callback function is executed after the current effect is finished.
- Typical syntax: `$(selector).hide(speed,callback);`
- **Examples**
- The example below has a callback parameter that is a function that will be executed after the hide effect is completed:





```
Eg with callback- <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/
  jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide("slow", function(){
      alert("The paragraph is now hidden");
    });
  });
});
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little content.</p>
</body>
```



- The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed

- **Example without Callback**

```
<script  
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>
```

```
$(document).ready(function(){  
  $("button").click(function(){  
    $("p").hide(1000);  
    alert("The paragraph is now hidden");  
  });  
});
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<button>Hide</button>
```

```
<p>This is a paragraph with little content.</p>
```

```
</body>
```



# JQUERY CHAINING

- With jQuery, we can chain together actions/methods.
- Until now we have been writing jQuery statements one at a time (one after the other).
- However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).
- That is Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.
- To chain an action, you simply append the action to the previous action.
- The following example chains together the `css()`, `slideUp()`, and `slideDown()` methods. The "p1" element first changes to red, then it slides up, and then it slides down:



Eg- `<script  
 src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>`

`<script>`

`$(document).ready(function(){`

`$("#button").click(function(){`

`$("#p1").css("color", "red").slideUp(2000).slideDown(2000);`

`});`

`});`

`</script>`

`</head>`

`<body>`

`<p id="p1">jQuery is fun!!</p>`

`<button>Click me</button>`

`</body>`



# JQUERY GET AND SET CONTENT AND ATTRIBUTES

- jQuery contains powerful methods for changing and manipulating HTML elements and attributes.
- One very important part of jQuery is the possibility to manipulate the DOM.

## **Get Content - text(), html(), and val()**

- Three simple, but useful, jQuery methods for DOM manipulation are:
- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.m
  in.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    alert("Text: " + $("#test").text());
  });
  $("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
  });
});
</script>
</head>
<body>
<p id="test">This is some <b>bold</b> text in a paragraph.</p>
<button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>
</body>
```



```
Eg <script  
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquer  
  y.min.js"></script>
```

```
<script>
```

```
$(document).ready(function(){  
  $("button").click(function(){  
    alert("Value: " + $("#test").val());  
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>Name: <input type="text" id="test" value="Mickey  
  Mouse"></p>
```

```
<button>Show Value</button>
```

```
</body>
```



## Get Attributes - attr()

- The jQuery attr() method is used to get attribute values.
- The following example demonstrates how to get the value of the href attribute in a link:

**Eg** `<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>`

```
<script>  
$(document).ready(function(){  
  $("#button").click(function(){  
    alert($("#w3s").attr("href"));  
  });  
});  
</script>  
</head>  
<body>  
<p><a href="https://www.w3schools.com"  
  id="w3s">W3Schools.com</a></p>  
<button>Show href Value</button>  
</body>
```





## Set Content - text(), html(), and val()

- We will use the same three methods from the previous page to **set content**:
- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields
- The following example demonstrates how to set content with the jQuery text(), html(), and val() methods:



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
  </script>
<script>$(document).ready(function(){
  $("#btn1").click(function(){
    $("#test1").text("Hello world!");
  });
  $("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
  });
  $("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
  });
});</script>
<body>
<p id="test1">This is a paragraph.</p>
<p id="test2">This is another paragraph.</p>
<p>Input field: <input type="text" id="test3" value="Mickey
  Mouse"></p>
<button id="btn1">Set Text</button>
<button id="btn2">Set HTML</button>
<button id="btn3">Set Value</button>
</body>
```



## Set Attributes - attr()

- The jQuery attr() method is also used to set/change attribute values.
- The following example demonstrates how to change (set) the value of the href attribute in a link:

### Eg

```
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#w3s").attr("href", "https://www.w3schools.com/jquery/");
  });
});
</script>
</head>
<body>
<p><a href="https://www.w3schools.com"
  id="w3s">W3Schools.com</a></p>
<button>Change href Value</button>
<p>Mouse over the link (or click on it) to see that the value of the href
  attribute has changed.</p>
</body>
```



# JQUERY –ADD ELEMENTS

- With jQuery, it is easy to add new elements/content.
- There are four jQuery methods that are used to add new content:
  1. `append()` - Inserts content at the end of the selected elements
  2. `prepend()` - Inserts content at the beginning of the selected elements
  3. `after()` - Inserts content after the selected elements
  4. `before()` - Inserts content before the selected elements



## 1. jQuery append() Method

- The jQuery append() method inserts content AT THE END of the selected HTML elements.

**Eg** -<script

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>$(document).ready(function(){  
    $("#btn1").click(function(){  
        $("p").append(" <b>Appended text</b>.");  
    });  
    $("#btn2").click(function(){  
        $("ol").append("<li>Appended item</li>");  
    });  
});</script>
```

```
<body>
```

```
<p>This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

```
<ol>
```

```
<li>List item 1</li>
```

```
<li>List item 2</li>
```

```
<li>List item 3</li>
```

```
</ol>
```

```
<button id="btn1">Append text</button>
```

```
<button id="btn2">Append list items</button>
```

```
</body>
```



## 2. jQuery prepend() Method

- The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

**Eg** `<script`

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>$(document).ready(function(){  
  $("#btn1").click(function(){  
    $("p").prepend("<b>Prepended text</b>. ");  
  });  
  $("#btn2").click(function(){  
    $("ol").prepend("<li>Prepended item</li>");  
  });  
});</script>
```

```
<body>
```

```
<p>This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

```
<ol>
```

```
<li>List item 1</li>
```

```
<li>List item 2</li>
```

```
<li>List item 3</li>
```

```
</ol>
```

```
<button id="btn1">Prepend text</button>
```

```
<button id="btn2">Prepend list item</button>
```

```
</body>
```



### 3. jQuery after() and before() Methods

- The jQuery after() method inserts content AFTER the selected HTML elements.
- The jQuery before() method inserts content BEFORE the selected HTML elements.

Eg-`<script`

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>
```

```
$(document).ready(function(){
```

```
  $("#btn1").click(function(){
```

```
    $("#img").before("<b>Before</b>");
```

```
  });
```

```
  $("#btn2").click(function(){
```

```
    $("#img").after("<i>After</i>");
```

```
  });
```

```
});
```

```
</script>
```

```
<body>
```

```
<br><br>
```

```
<button id="btn1">Insert before</button>
```

```
<button id="btn2">Insert after</button>
```

```
</body>
```



# ADD SEVERAL NEW ELEMENTS

## 1. Add Several New Elements With `append()` and `prepend()`

- However, both `append()` and `prepend()` methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML, with jQuery, or with JavaScript code and DOM elements.





```
Eg-<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/
  jquery.min.js"></script>
<script>
function appendText() {
  var txt1 = "<p>Text.</p>";    // Create text with HTML
  var txt2 = $("<p></p>").text("Text."); // Create text with
  jQuery
  var txt3 = document.createElement("p");
  txt3.innerHTML = "Text.";    // Create text with DOM
  $("body").append(txt1, txt2, txt3); // Append new
  elements
}
</script>
</head>
<body>
<p>This is a paragraph.</p>
<button onclick="appendText()">Append text</button>
</body>
```



## 2. Add Several New Elements With `after()` and `before()`

- Also, both the `after()` and `before()` methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML, with jQuery, or with JavaScript code and DOM elements.



```
Eg <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.m
  in.js"></script>
<script>
function afterText() {
  var txt1 = "<b>I </b>";      // Create element with HTML
  var txt2 = $("<i></i>").text("love "); // Create with jQuery
  var txt3 = document.createElement("b"); // Create with DOM
  txt3.innerHTML = "jQuery!";
  $("img").after(txt1, txt2, txt3); // Insert new elements after
  img
}
</script>
</head>
<body>

<p>Click the button to insert text after the image.</p>
<button onclick="afterText()">Insert after</button>
</body>
```



# JQUERY REMOVE ELEMENTS

- With jQuery, it is easy to remove existing HTML elements.
- To remove elements and content, there are mainly two jQuery methods:
- `remove()` - Removes the selected element (and its child elements)
- `empty()` - Removes the child elements from the selected element
- **jQuery `remove()` Method**
- The jQuery `remove()` method removes the selected element(s) and its child elements.



Eg `<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.m  
in.js"></script>`

`<script>`

`$(document).ready(function(){`

`$("#button").click(function(){`

`$("#div1").remove();`

`});`

`});`

`</script>`

`<body>`

`<div id="div1" style="height:100px;width:300px;border:1px solid  
black;background-color:yellow;">`

This is some text in the div.

`<p>This is a paragraph in the div.</p>`

`<p>This is another paragraph in the div.</p>`

`</div>`

`<br>`

`<button>Remove div element</button>`

`</body>`



- **jQuery empty() Method**

- The jQuery empty() method removes the child elements of the selected element(s).

Eg-<script

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>
```

```
$(document).ready(function(){
```

```
  $("button").click(function(){
```

```
    $("#div1").empty();
```

```
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<div id="div1" style="height:100px;width:300px;border:1px solid  
  black;background-color:yellow;">
```

```
This is some text in the div.
```

```
<p>This is a paragraph in the div.</p>
```

```
<p>This is another paragraph in the div.</p>
```

```
</div>
```

```
<br>
```

```
<button>Empty the div element</button>
```

```
</body>
```



# JQUERY GET AND SET CSS CLASSES

- With jQuery, it is easy to manipulate the style of elements.
- jQuery has several methods for CSS manipulation. We will look at the following methods:
- `addClass()` - Adds one or more classes to the selected elements
- `removeClass()` - Removes one or more classes from the selected elements
- `toggleClass()` - Toggles between adding/removing classes from the selected elements
- `css()` - Sets or returns the style attribute



## jQuery `addClass()` Method

The following example shows how to add class attributes to different elements. Of course you can select multiple elements, when adding classes:





```
Eg-<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
  });
});
</script>
<style>
.important {
  font-weight: bold;
  font-size: xx-large;
}
.blue {
  color: blue;
}
</style>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<div>This is some important text!</div><br>
<button>Add classes to elements</button>
</body>
```



## jQuery removeClass() Method

- The following example shows how to remove a specific class attribute from different elements:

Eg `<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>`

```
<script>
```

```
$(document).ready(function(){  
  $("#button").click(function(){  
    $("#h1, h2, p").removeClass("blue");  
  });
```

```
});
```

```
</script>
```

```
<style>
```

```
.important {  
  font-weight: bold;  
  font-size: xx-large;  
}
```

```
.blue {  
  color: blue;  
}
```

```
</style>
```

```
<body>
```

```
<h1 class="blue">Heading 1</h1>
```

```
<h2 class="blue">Heading 2</h2>
```

```
<p class="blue">This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

```
<button>Remove class from elements</button>
```

```
</body>
```



## jQuery toggleClass() Method

- The following example will show how to use the jQuery toggleClass() method. This method toggles between adding/removing classes from the selected elements:



Eg- `<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">  
</script>`

`<script>`

```
$(document).ready(function(){  
  $("button").click(function(){  
    $("h1, h2, p").toggleClass("blue");  
  });  
});
```

`</script>`

`<style>`

`.blue {`

`color: blue;`

`}`

`</style>`

`<body>`

`<h1>Heading 1</h1>`

`<h2>Heading 2</h2>`

`<p>This is a paragraph.</p>`

`<p>This is another paragraph.</p>`

`<button>Toggle class</button>`

`</body>`



# JQUERY CSS() METHOD

## **jQuery css() Method**

- The `css()` method sets or returns one or more style properties for the selected elements.
- Return a CSS Property
- To return the value of a specified CSS property, use the following syntax:
- `css("propertyname");`



Eg. `<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.  
min.js"></script>`

`<script>`

`$(document).ready(function(){`

`$("#button").click(function(){`

`alert("Background color = " + $("#p").css("background-  
color"));`

`});`

`});`

`</script>`

`<body>`

`<h2>This is a heading</h2>`

`<p style="background-color:#ff0000">This is a paragraph.</p>`

`<p style="background-color:#00ff00">This is a paragraph.</p>`

`<p style="background-color:#0000ff">This is a paragraph.</p>`

`<button>Return background-color of p</button>`

`</body>`



## Set a CSS Property

- To set a specified CSS property, use the following syntax:
- `css("propertyname","value");`

**Eg** `<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">  
</script>`

```
<script>
```

```
$(document).ready(function(){  
  $("button").click(function(){  
    $("#p").css("background-color", "yellow");  
  });  
});
```

```
</script>
```

```
<body>
```

```
<h2>This is a heading</h2>
```

```
<p style="background-color:#ff0000">This is a paragraph.</p>
```

```
<p style="background-color:#00ff00">This is a paragraph.</p>
```

```
<p style="background-color:#0000ff">This is a paragraph.</p>
```

```
<p>This is a paragraph.</p>
```

```
<button>Set background-color of p</button>
```

```
</body>
```



## Set Multiple CSS Properties

- To set multiple CSS properties, use the following syntax:
- `css({"propertyname":"value","propertyname":"value",...});`

### Eg

- `$("p").css({"background-color": "yellow", "font-size": "200%"});`





# JQUERY NOCONFLICT() METHOD

- jQuery uses the \$ sign as a shortcut for jQuery.
- There are many other popular JavaScript frameworks like: Angular, Backbone, Ember, Knockout, and more.
- If two different frameworks are using the same shortcut, one of them might stop working.
- The jQuery team have already thought about this, and implemented the noConflict() method.
- The noConflict() method releases the hold on the \$ shortcut identifier, so that other scripts can use it.



Eg

```
<script  
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script>
```

```
$.noConflict();
```

```
jQuery(document).ready(function(){
```

```
  jQuery("button").click(function(){
```

```
    jQuery("p").text("jQuery is still working!");
```

```
  });
```

```
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p>This is a paragraph.</p>
```

```
<button>Test jQuery</button>
```

```
</body>
```



# REFERENCES

<https://www.tutorialspoint.com/jquery>

<https://www.w3schools.com/jquery/>

<https://www.javatpoint.com/jquery-tutorial>



The background of the slide is a photograph of a modern, multi-story building with a light-colored facade and dark window frames. The building has a prominent entrance with a blue archway. The text "Mangalochkar Institute of Management" is visible on the building's facade. The sky is blue with some light clouds.

**Thank You !!!**

**Any Queries**

**7721082286/shrushti.rapelli@gmail.com**

# Digital Electronics

## Logic Gates

By  
Mrs. Deshpande M.K.

# Logic Gates

- **The logic gates are the basic building blocks of digital electronics.**
- **Logic gate is a digital circuit follows a logical relationship between the input and output.**
- **Logic gates are the switches that turn ON or OFF depending on what the user is doing**
- **Logic gates turn ON when a certain condition is true, and OFF when the condition is false**

# Basic Gates

All digital systems can be constructed by only three basic logic gates. These basic gates are called the **AND gate**, the **OR gate**, and the **NOT gate**.

**Basic Gates**

**NOT**

$X \rightarrow \text{NOT} \rightarrow Y$

$Y = \neg X$   
 $Y = \text{not } X$   
 $Y = \sim X$   
 $Y = X'$

X	Y
0	1
1	0

**AND**

$X, Y \rightarrow \text{AND} \rightarrow Z$

$Z = X \& Y$   
 $Z = X \text{ and } Y$   
 $Z = X * Y$   
 $Z = XY$

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

**OR**

$X, Y \rightarrow \text{OR} \rightarrow Z$

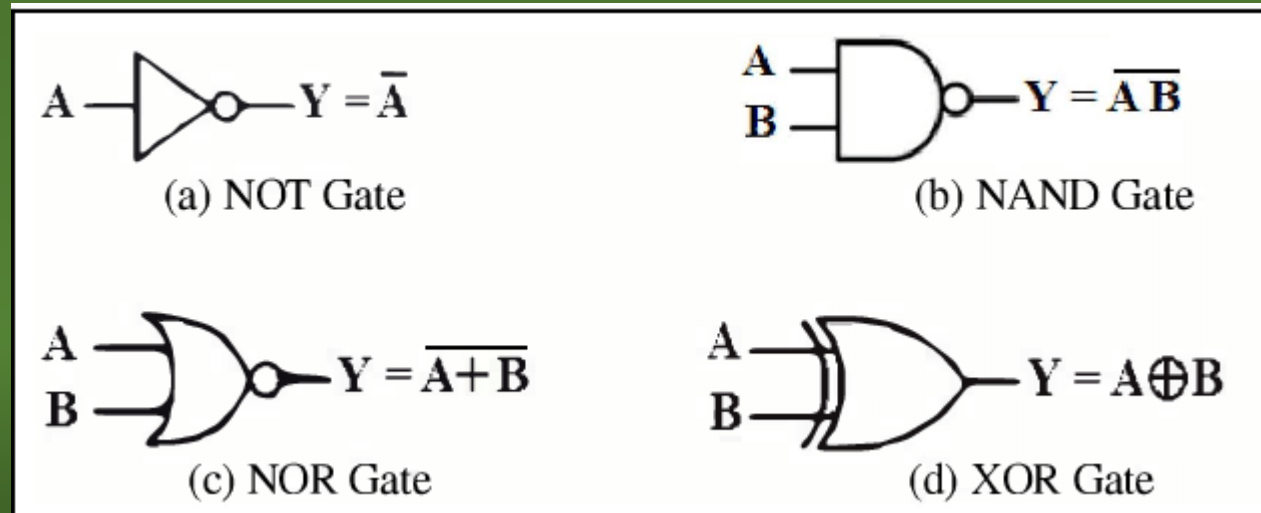
$Z = X \# Y$   
 $Z = X \text{ or } Y$   
 $Z = X \mid Y$   
 $Z = X + Y$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

**Any logic circuit can be created using only these three gates**

# Derived Gates

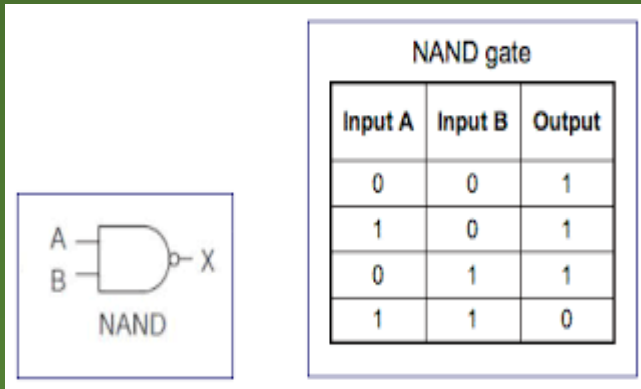
The logic gates which are derived from the basic gates such as **AND**, **OR**, **NOT** gates are called derived gates. These derived gates have their own unique Symbols, Truth Tables and Boolean Expressions. Here we will explore the most common derived gates such as NAND Gate, NOR Gate, EX-OR Gate, and EX-NOR Gate.



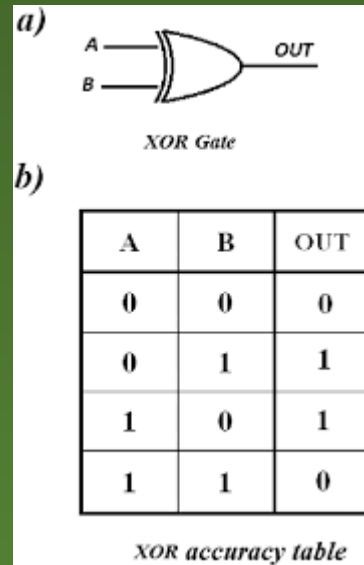


# Derived Gates with Truth Table

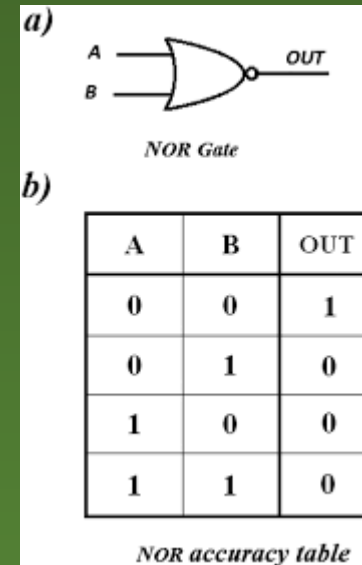
## NAND



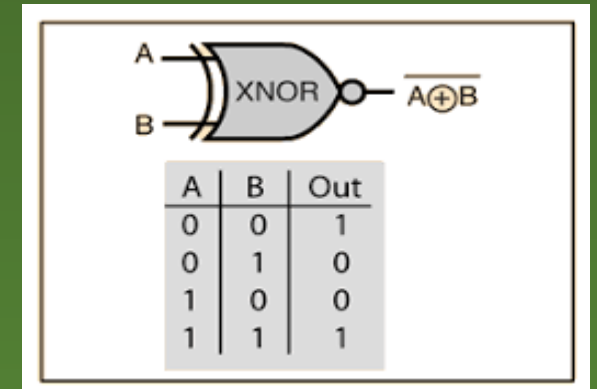
## XOR



## NOR



## XNOR



**Thank you.**

Faculty-Nilima Mondhe  
Class-BCA I  
Sem-II  
Subject-Advance web Programming

## What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

## Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

## History of JavaScript

In 1993, **Mosaic**, the first popular web browser, came into existence. In the **year 1994, Netscape** was founded by **Marc Andreessen**. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently, in

1995, the company recruited **Brendan Eich** intending to implement and embed Scheme programming language to the browser. But, before Brendan could start, the company merged with **Sun Microsystems** for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Now, two languages were there: Java and the scripting language. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'Javascript'. Finally, in May 1995, Marc Andreessen coined the first code of Javascript named '**Mocha**'. Later, the marketing team replaced the name with '**LiveScript**'. But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

## Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

## JavaScript Example

```
[OBJ]  
[OBJ]
```

1. `<script>`
2. `document.write("Hello JavaScript by JavaScript");`
3. `</script>`

### Test it Now

A detailed explanation of first JavaScript example is given in next chapter.

1. `script type="text/javascript" >`
2. `document.write("JavaScript is a simple language for javatpoint learners");`
3. `</script>`

The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

## Stay

The **document.write()** function is used to display dynamic content through JavaScript. We will learn about document object in detail later.

### 3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)



#### 1) JavaScript Example : code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

1. `<script type="text/javascript">`
2. `alert("Hello Javatpoint");`
3. `</script>`

#### 2) JavaScript Example : code between the head tag

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

1. `<html>`
2. `<head>`
3. `<script type="text/javascript">`
4. `function msg(){`
5. `alert("Hello Javatpoint");`
6. `}`
7. `</script>`
8. `</head>`
9. `<body>`
10. `<p>Welcome to JavaScript</p>`
11. `<form>`
12. `<input type="button" value="click" onclick="msg()"/>`

13. `</form>`
14. `</body>`
15. `</html>`

## JavaScript Comment

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

### *Advantages of JavaScript comments*

There are mainly two advantages of JavaScript comments.

## JavaScript Variable

1. [JavaScript variable](#)
2. [JavaScript Local variable](#)
3. [JavaScript Global variable](#)

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( \$ ) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

### Correct JavaScript variables

1. `var x = 10;`
2. `var _value="sonoo";`

### Incorrect JavaScript variables

1. `var 123=30;`
2. `var *aa=320;`

## Example of JavaScript variable

Let's see a simple example of JavaScript variable.

Competitive questions on Structures in HindiKeep Watching

1. `<script>`
2. `var x = 10;`
3. `var y = 20;`
4. `var z=x+y;`
5. `document.write(z);`

6. `</script>`

*Output of the above example*

30

## JavaScript local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

```
1. <script>  
2. function abc(){  
3. var x=10;//local variable  
4. }  
5. </script>
```

Or,

```
1. <script>  
2. if(10<13){  
3. var y=20;//JavaScript local variable  
4. }  
5. </script>
```

## JavaScript global variable

A JavaScript global variable is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

```
1. <script>  
2. var data=200;//global variable  
3. function a(){  
4. document.writeln(data);  
5. }  
6. function b(){  
7. document.writeln(data);  
8. }  
9. a();//calling JavaScript function  
10. b();  
11. </script>
```

## JavaScript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

1. `var a=40;//holding number`
2. `var b="Rahul";//holding string`
- 3.

## JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

## JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

We will have great discussion on each data type later.

## JavaScript If-else

The **JavaScript if-else statement** is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

## JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

1. `if(expression){`
2. `//content to be evaluated`



3.            }

## Flowchart of JavaScript If statement

Let's see the simple example of if statement in javascript.

```
1.            <script>
2.            var a=20;
3.            if(a>10){
4.            document.write("value of a is greater than 10");
5.            }
6.            </script>
```

## JavaScript Loops

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

### 1) JavaScript For loop

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

```
1.            for (initialization; condition; increment)
2.            {
3.            code to be executed
4.            }
```

Let's see the simple example of for loop in javascript.

```
1.            <script>
2.            for (i=1; i<=5; i++)
3.            {
4.            document.write(i + "<br/>")
5.            }
6.            </script>
```

Output:

```
1
2
3
4
5
```

## 2) JavaScript while loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known. The syntax of while loop is given below.

1. while (condition)
2. {
3. code to be executed
4. }

Let's see the simple example of while loop in javascript.

1. <script>
2. var i=11;
3. while (i<=15)
4. {
5. document.write(i + "<br/>");
6. i++;
7. }
8. </script>

### **Test it Now**

Output:

```
11
12
13
14
15
```

## 3) JavaScript do while loop

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least once* whether condition is true or false. The syntax of do while loop is given below.

1. do{
2. code to be executed
3. }while (condition);

Let's see the simple example of do while loop in javascript.

1. <script>
2. var i=21;
3. do{
4. document.write(i + "<br/>");
5. i++;
6. }while (i<=25);
7. </script>

### **Test it Now**

Output:

21  
22  
23  
24  
25

## 4) JavaScript for in loop

The **JavaScript for in loop** is used to *iterate the properties of an object*. We will discuss about it later.

## JavaScript Switch

The **JavaScript switch statement** is used to *execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

```
1.      switch(expression){
2.      case value1:
3.      code to be executed;
4.      break;
5.      case value2:
6.      code to be executed;
7.      break;
8.      .....
9.
10.     default:
11.     code to be executed if above values are not matched;
12.     }
```

Let's see the simple example of switch statement in javascript.

```
1.      <script>
2.      var grade='B';
3.      var result;
4.      switch(grade){
5.      case 'A':
6.      result="A Grade";
7.      break;
8.      case 'B':
9.      result="B Grade";
10.     break;
11.     case 'C':
12.     result="C Grade";
13.     break;
14.     default:
15.     result="No Grade";
16.     }
```

17. `document.write(result);`
18. `</script>`

### *Output of the above example*

B Grade

*The switch statement is fall-through i.e. all the cases will be evaluated if you don't use break statement.*

Let's understand the behaviour of switch statement in JavaScript.

1

1. `<script>`
2. `var grade='B';`
3. `var result;`
4. `switch(grade){`
5. `case 'A':`
6. `result+=" A Grade";`
7. `case 'B':`
8. `result+=" B Grade";`
9. `case 'C':`
10. `result+=" C Grade";`
11. `default:`
12. `result+=" No Grade";`
13. `}`
14. `document.write(result);`
15. `</script>`

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

### *Advantage of JavaScript function*

There are mainly two advantages of JavaScript functions.

1. **Code reusability:** We can call a function several times so it save coding.
2. **Less coding:** It makes our program compact. We don't need to write many lines of code each time to perform a common task.

## JavaScript Function Syntax

The syntax of declaring function is given below.

1. `function functionName([arg1, arg2, ...argN]){`
2. `//code to be executed`
3. `}`

JavaScript Functions can have 0 or more arguments.

## JavaScript Function Example

Let's see the simple example of function in JavaScript that does not has arguments.

```
1. <script>
2.   function msg(){
3.     alert("hello! this is message");
4.   }
5. </script>
6. <input type="button" onclick="msg()" value="call function"/>
```

*Output of the above example*

## JavaScript Function Arguments

We can call function by passing arguments. Let's see the example of function that has one argument.

```
1. <script>
2.   function getcube(number){
3.     alert(number*number*number);
4.   }
5. </script>
6. <form>
7.   <input type="button" value="click" onclick="getcube(4)"/>
8. </form>
```

### **Test it Now**

*Output of the above example*

Top of Form  
Bottom of Form

## Function with Return Value

We can call function that returns a value and use it in our program. Let's see the example of function that returns value.

```
1. <script>
2.   function getInfo(){
3.     return "hello javatpoint! How r u?";
4.   }
5. </script>
6. <script>
7.   document.write(getInfo());
8. </script>
```

## JavaScript Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the

execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

**For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

### Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

### Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown onkeyup	&When the user press and then release the key

### Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

### Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Let's discuss some examples over events and their handlers.

## Click Event

```
1. <html>
2. <head> Javascript Events </head>
3. <body>
4. <script language="Javascript" type="text/Javascript">
5. <!--
6.     function clickevent()
7.     {
8.         document.write("This is JavaTpoint");
9.     }
10. //-->
11. </script>
12. <form>
13. <input type="button" onclick="clickevent()" value="Who's this?"/>
14. </form>
15. </body>
16. </html>
```

## MouseOver Event

```
1. <html>
2. <head>
3. <h1> Javascript Events </h1>
4. </head>
5. <body>
6. <script language="Javascript" type="text/Javascript">
7. <!--
8.     function mouseoverevent()
9.     {
10.         alert("This is JavaTpoint");
11.     }
12. //-->
13. </script>
14. <p onmouseover="mouseoverevent()"> Keep cursor over me </p>
15. </body>
16. </html>
```

## Focus Event

```
1. <html>
2. <head> Javascript Events </head>
3. <body>
4. <h2> Enter something here </h2>
5. <input type="text" id="input1" onfocus="focusevent()"/>
6. <script>
```

```

7.      <!--
8.      function focusevent()
9.      {
10.     document.getElementById("input1").style.background=" aqua";
11.     }
12.     //-->
13.     </script>
14.     </body>
15.     </html>

```

## Keydown Event

```

1.     <html>
2.     <head> Javascript Events </head>
3.     <body>
4.     <h2> Enter something here </h2>
5.     <input type="text" id="input1" onkeydown="keydownevent()" />
6.     <script>
7.     <!--
8.     function keydownevent()
9.     {
10.    document.getElementById("input1");
11.    alert("Pressed a key");
12.    }
13.    //-->
14.    </script>
15.    </body>
16.    </html>

```

## Load event

```

1.     <html>
2.     <head> Javascript Events </head>
3.     </br>

```

<body **onload**="window.alert" **JavaScript Array**

**JavaScript array** is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

### 1) JavaScript array literal

The syntax of creating array using array literal is given below:

```

1.     var arrayname=[value1,value2.....valueN];

```



As you can see, values are contained inside [ ] and separated by , (comma).  
Let's see the simple example of creating and using array in JavaScript.

```
1.     <script>
2.     var emp=["Sonoo","Vimal","Ratan"];
3.     for(i=0;i<emp.length;i++){
4.     document.write(emp[i] + "<br/>");
5.     }
6.     </script>
```

The .length property returns the length of an array.

### Output of the above example

```
Sonoo
Vimal
Ratan
```

## 2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
1.     var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array.  
Let's see the example of creating array directly.

```
1.     <script>
2.     var i;
3.     var emp = new Array();
4.     emp[0] = "Arun";
5.     emp[1] = "Varun";
6.     emp[2] = "John";
7.
8.     for(i=0;i<emp.length;i++){
9.     document.write(emp[i] + "<br>");
10.    }
11.    </script>
```

### Output of the above example

```
Arun
Varun
John
```

## 3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

```
1.     <script>
2.     var emp=new Array("Jai","Vijay","Smith");
3.     for(i=0;i<emp.length;i++){
4.     document.write(emp[i] + "<br>");
```

5.        }
6.        </script>

### Output of the above example

```
Jai  
Vijay  
Smith
```

## JavaScript Array Methods

Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value is an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.

lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

```

4.     ('Page successfully loaded');" >
5.     <script>
6.     <!--
7.     document.write("The page is loaded successfully");
8.     //-->
9.     </script>
10.    </body>
11.    </html>

```

## JavaScript String

The **JavaScript string** is an object that represents a sequence of characters. There are 2 ways to create string in JavaScript

1. By string literal

2. By string object (using new keyword)



## 1) By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

1. `var stringname="string value";`

Let's see the simple example of creating string literal.

Competitive questions on Structures in HindiKeep Watching

1. `<script>`
2. `var str="This is string literal";`
3. `document.write(str);`
4. `</script>`

### Output:

```
This is string literal
```



## 2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

1. `var stringname=new String("string literal");`

Here, **new keyword** is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

1. `<script>`
2. `var stringname=new String("hello javascript string");`
3. `document.write(stringname);`
4. `</script>`

### Output:

```
hello javascript string
```



## JavaScript String Methods

Let's see the list of JavaScript string methods with examples.

Methods	Description
<code>charAt()</code>	It provides the char value present at the specified index.
<code>charCodeAt()</code>	It provides the Unicode value of a character present at the specified index.
<code>concat()</code>	It provides a combination of two or more strings.
<code>indexOf()</code>	It provides the position of a char value present in the given string.
<code>lastIndexOf()</code>	It provides the position of a char value present in the given string by searching a character from the last position.

search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.



## 1) JavaScript String charAt(index) Method

The JavaScript String charAt() method returns the character at the given index.

1. `<script>`
2. `var str="javascript";`
3. `document.write(str.charAt(2));`
4. `</script>`

### Output:



## 2) JavaScript String concat(str) Method

The JavaScript String concat(str) method concatenates or joins two strings.

1. `<script>`
2. `var s1="javascript";`
3. `var s2="concat example";`
4. `var s3=s1.concat(s2);`
5. `document.write(s3);`
6. `</script>`

**Output:**

```
javascript concat example
```

### 3) JavaScript String indexOf(str) Method

The JavaScript String indexOf(str) method returns the index position of the given string.

```
1.     <script>
2.     var s1="javascript from javatpoint indexof";
3.     var n=s1.indexOf("from");
4.     document.write(n);
5.     </script>
```

**Output:**

```
11
```

### 4) JavaScript String lastIndexOf(str) Method

The JavaScript String lastIndexOf(str) method returns the last index position of the given string.

```
1.     <script>
2.     var s1="javascript from javatpoint indexof";
3.     var n=s1.lastIndexOf("java");
4.     document.write(n);
5.     </script>
```

**Output:**

```
16
```

### 5) JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters.

```
1.     <script>
2.     var s1="JavaScript toLowerCase Example";
3.     var s2=s1.toLowerCase();
4.     document.write(s2);
5.     </script>
```

**Output:**

```
javascript tolowercase example
```

### 6) JavaScript String toUpperCase() Method

The JavaScript String toUpperCase() method returns the given string in uppercase letters.

```
1.     <script>
2.     var s1="JavaScript toUpperCase Example";
3.     var s2=s1.toUpperCase();
4.     document.write(s2);
```

5. `</script>`

#### Output:

```
JAVASCRIPT TOUPPERCASE EXAMPLE
```

### 7) JavaScript String slice(beginIndex, endIndex) Method

The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

```
1. <script>
2.   var s1="abcdefgh";
3.   var s2=s1.slice(2,5);
4.   document.write(s2);
5. </script>
```

#### Output:

```
cde
```

### 8) JavaScript String trim() Method

The JavaScript String trim() method removes leading and trailing whitespaces from the string.

```
1. <script>
2.   var s1=" javascript trim ";
3.   var s2=s1.trim();
4.   document.write(s2);
5. </script>
```

## JavaScript Form Validation

1. [JavaScript form validation](#)
2. [Example of JavaScript validation](#)
3. [JavaScript email validation](#)

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window. So

1. window.document  
Is same as

## Competitive questions on Structures in HindiKeep Watching

### 1. document

According to W3C - "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

## Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

## Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the document.
writeln("string")	writes the given string on the document with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

## Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.

Here, **document** is the root element that represents the html document.

**form1** is the name of the form.

**name** is the attribute name of the input text.

**value** is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

1. <script type="text/javascript">
2. function printvalue(){
3. var name=document.form1.name.value;



```

4.     alert("Welcome:" + name);
5.     }
6.     </script>
7.     -
8.     <form name="form1">
9.     Enter Name:<input type="text" name="name"/>
10.    <input type="button" onclick="printvalue()" value="print name"/>
11.    </form>

```

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.



## JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```

1.     <script>
2.     function validateform(){
3.     var name=document.myform.name.value;
4.     var password=document.myform.password.value;
5.
6.     if (name==null || name==""){
7.     alert("Name can't be blank");
8.     return false;
9.     }else if(password.length<6){
10.    alert("Password must be at least 6 characters long.");
11.    return false;
12.    }
13.    }
14.    </script>
15.    <body>
16.    <form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()
">
17.    Name: <input type="text" name="name"> <br/>
18.    Password: <input type="password" name="password"> <br/>
19.    <input type="submit" value="register">
20.    </form>

```



## JavaScript Retype Password Validation

```
1.     <script type="text/javascript">
2.     function matchpass(){
3.     var firstpassword=document.f1.password.value;
4.     var secondpassword=document.f1.password2.value;
5.     -
6.     if(firstpassword==secondpassword){
7.     return true;
8.     }
9.     else{
10.    alert("password must be same!");
11.    return false;
12.    }
13.    }
14.    </script>
15.    -
16.    <form name="f1" action="register.jsp" onsubmit="return matchpass()">
17.    Password:<input type="password" name="password" /> <br/>
18.    Re-enter Password:<input type="password" name="password2" /> <br/>
19.    <input type="submit">
20.    </form>
```

### Test it Now



## JavaScript Number Validation

Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

```
1.     <script>
2.     function validate(){
3.     var num=document.myform.num.value;
4.     if(isNaN(num)){
5.     document.getElementById("numloc").innerHTML="Enter Numeric value only";
6.     return false;
7.     }else{
8.     return true;
9.     }
10.    }
11.    </script>
12.    <form name="myform" onsubmit="return validate()">
13.    Number:<input type="text" name="num"><span id="numloc"></span> <br/>
14.    <input type="submit" value="submit">
15.    </form>
```





**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of Management**  
**Career Development and Research.**

**156-B Railway Lines Solapur.**

**Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject Networking**

**Programme:- BCA II Sem III**

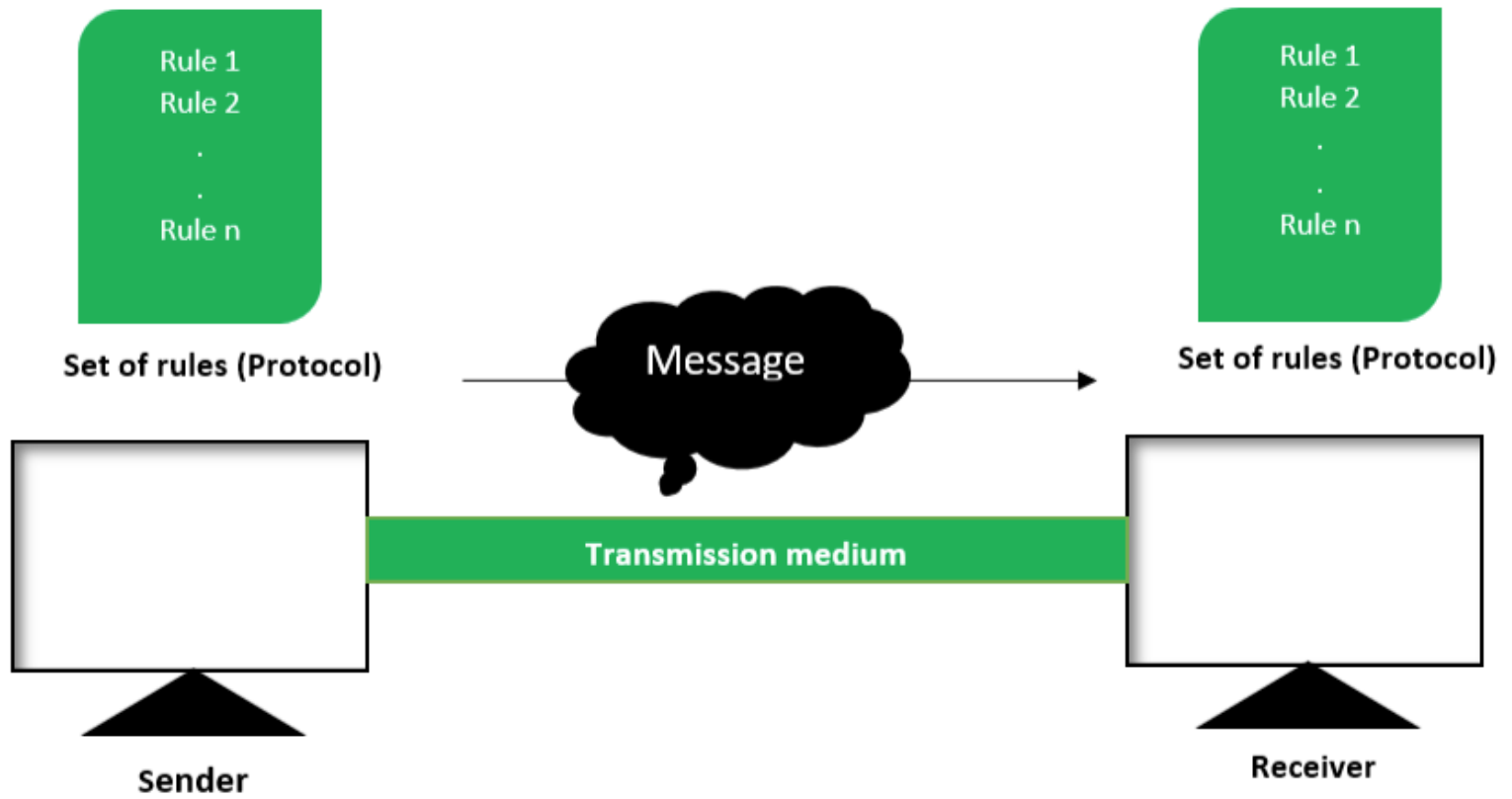
# Data representation

- **Data representation.** ... A binary digit, or bit, is the smallest unit of **data** in **computing**. It is **represented** by a 0 or a 1. Binary numbers are made up of binary digits (bits), eg the binary number 1001. The circuits in a **computer's** processor are made up of billions of transistors.

# Concepts of computer communication

- Communication is the process of transferring messages from one point to another
- It is defined as exchange of data between two devices via some form of transmission media such as a cable, wire or it can be air or vacuum also. For occurrence of data communication, communicating devices must be a part of communication system made up of a combination of hardware or software devices and programs.

- **Data Communication System Components :**  
There are mainly five components of a data communication system:
- **1. Message 2. Sender 3. Receiver 4. Transmission Medium 5. Set of rules (Protocol)**





- For Example When we speak to our friend on the telephone we are the sender, the telephone line, through which our voice is transmitted is the medium and friend is the receiver.. This is a simple example of voice communication. The same concept for data communication also. Data communication is the function of transporting data from one location to another. In this example the sender and receiver are normally machines for example computers ,transmission medium is telephone lines, satellites and messages that are transmitted are data Hence the electronic systems ,which transfer data from one location to another are called data communication system.

- **Message :**

This is most useful asset of a data communication system. The message simply refers to data or piece of information which is to be communicated. A message could be in any form, it may be in form of a text file, an audio file, a video file, etc.

- **Sender :**

To transfer message from source to destination, someone must be there who will play role of a source. Sender plays part of a source in data communication system. It is simple a device that sends data message. The device could be in form of a computer, mobile, telephone, laptop, video camera, or a workstation, etc.



- **Receiver :**

It is destination where finally message sent by source has arrived. It is a device that receives message. Same as sender, receiver can also be in form of a computer, telephone mobile, workstation, etc.

- 

- **Transmission Medium :**

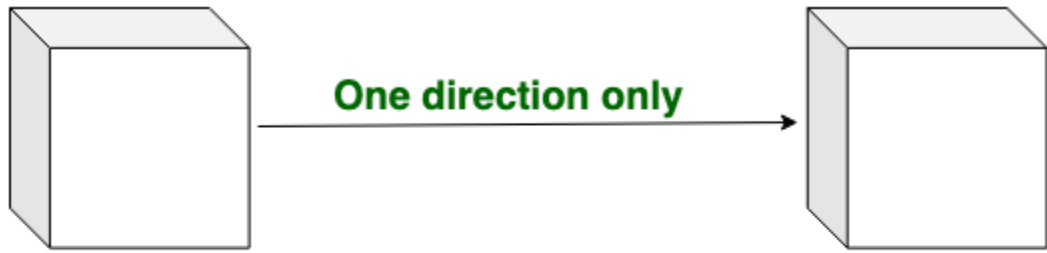
In entire process of data communication, there must be something which could act as a bridge between sender and receiver, Transmission medium plays that part. It is physical path by which data or message travels from sender to receiver. Transmission medium could be guided (with wires) or unguided (without wires), for example, twisted pair cable, fiber optic cable, radio waves, microwaves, etc.

- **Set of rules (Protocol) :**

To govern data communications, various sets of rules had been already designed by the designers of the communication systems, which represent a kind of agreement between communicating devices. These are defined as protocol. In simple terms, the protocol is a set of rules that govern data communication. If two different devices are connected but there is no protocol among them, there would not be any kind of communication between those two devices. Thus the protocol is necessary for data communication to take place.

# Data Transmission modes or Communication Components

- There are 3 types of transmission modes which are given below: Simplex mode, Half duplex mode, and Full duplex mode. These are explained as following below.
- **Simplex mode:**  
In simplex mode, Sender can send the data but that sender can't receive the data. It is a unidirectional communication.

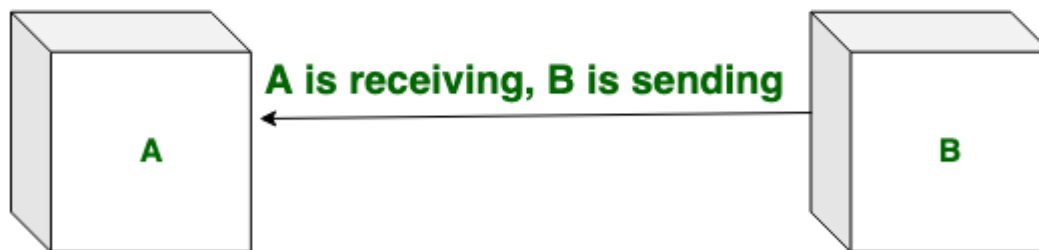
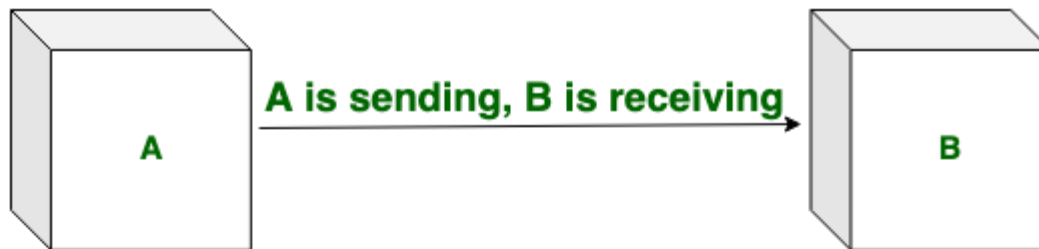


**Simplex mode**

- **Half-duplex mode:**

In half duplex mode, Sender can send the data and also can receive the data but one at a time. It is two-way directional communication but one at a time.

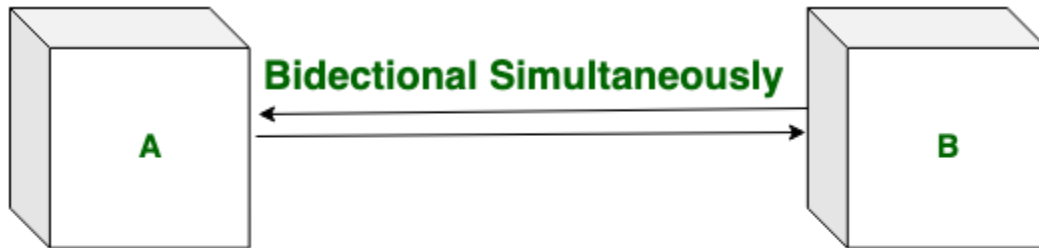




**Half duplex mode**

- **Full duplex mode:**

In full duplex mode, Sender can send the data and also can receive the data simultaneously. It is two-way directional communication simultaneously.



**Full duplex mode**

# Computer Network

- A computer network is a network of geographically distributed multiple computers connected to enable meaningful transmission and exchange of information among them. Sharing of information ,sharing of resources (both hardware and software) and sharing of processing load are some of the major objectives of a computer network

# Uses of Computer Network

- **Computer networks** support an enormous number of applications and services such as access to the World Wide Web, digital video, digital audio, shared **use** of application and storage servers, printers, and fax machines, and **use** of email and instant messaging applications as well as many others.

# network

## N/W Components:

- Hubs
- A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

-

- **Types of Hub**
- **Active Hub:-** These are the hubs which have their own power supply and can clean, boost and relay the signal along with the network. It serves both as a repeater as well as wiring centre. These are used to extend the maximum distance between nodes.
- **Passive Hub :-** These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can't be used to extend the distance between nodes.





# Switch

- **Switch** – A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.



# Repeater

- Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

## Repeater & Hub

### Functions, Advantages & Limitations



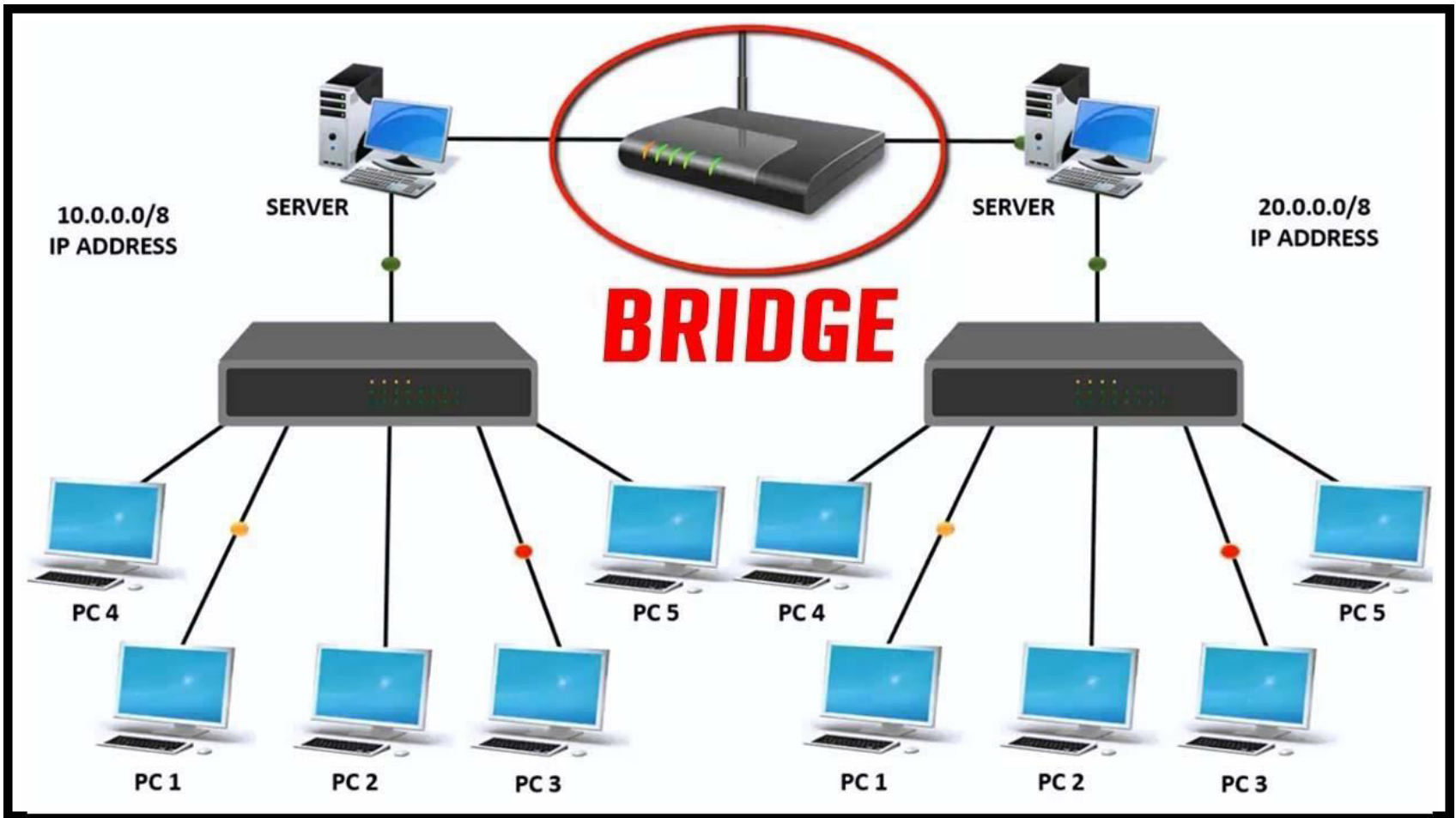
Repeater



Hub

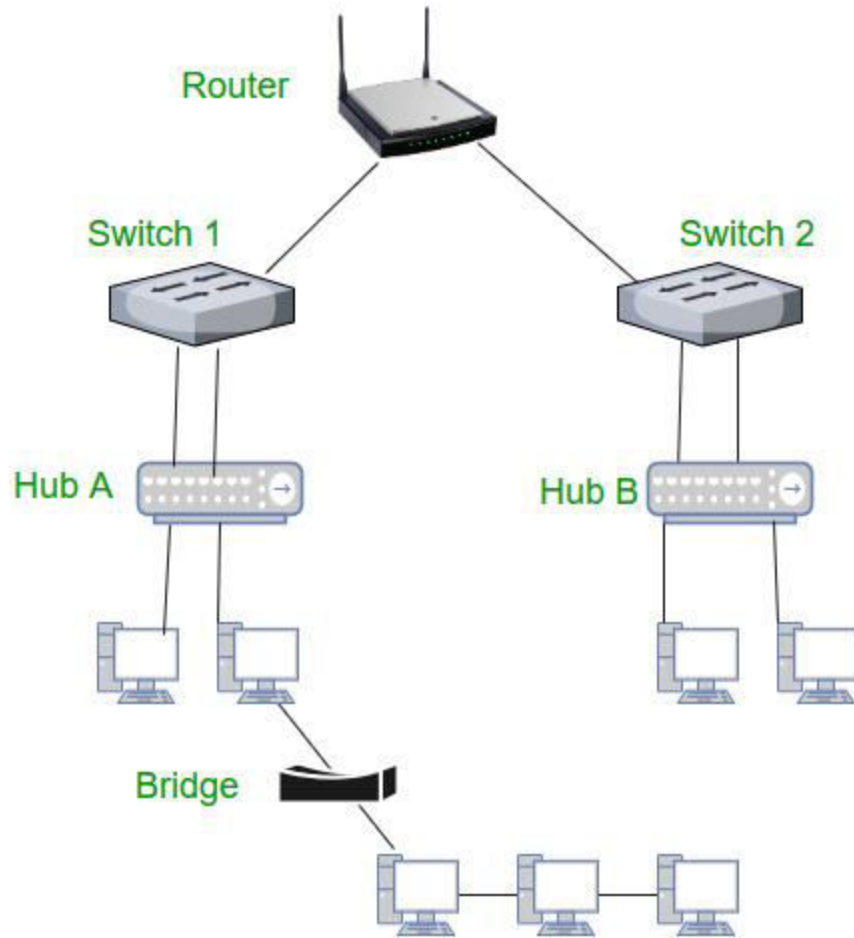
# Bridge

- Bridge – A bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.
- Types of Bridges
- Transparent Bridges:- These are the bridge in which the stations are completely unaware of the
- bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of
- the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.
- Source Routing Bridges:- In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The host can discover frame by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.



# Routers

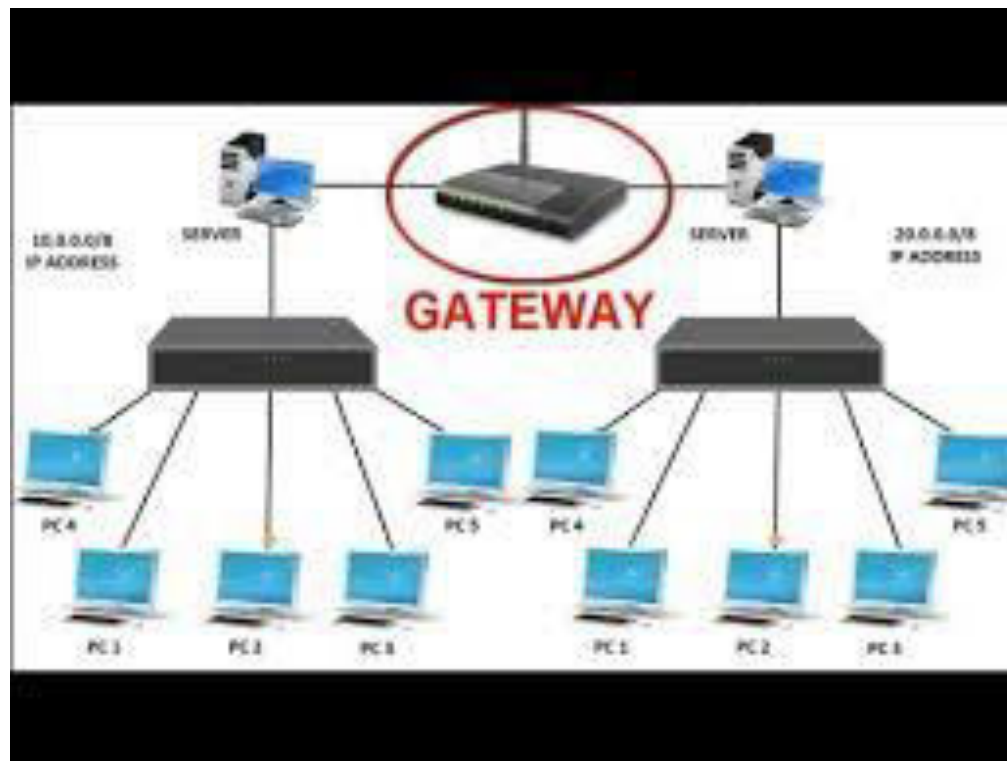
- – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.





# Gateway –

- A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.



# Applications of Internet

- Sending and receiving email
- Searching and browsing information archives
- Copying files between computers
- Conducting financial transactions
- Playing computer games
- Video and music streaming
- Chat or voice communication (direct messaging, video conferencing)

- Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the web that use Internet as a medium for communication. One can find various social networking sites such as:
  - Facebook
  - Twitter
  - Instagram
- One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.
- Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet.
  - Online Television jio tv
  - Online computer Games bike race car race, chess
  - Songs
  - Videos
  - Social Networking Apps
- Internet allows us to use many services like:
  - Internet Banking
  - Online Shopping
  - Online Ticket Booking
  - Online Bill Payment
  - Data Sharing
  - E-mail
- Internet provides concept of **electronic commerce**, that allows the business deals to be conducted on electronic systems

The background of the slide is a photograph of a modern, multi-story building with a light-colored facade and vertical architectural elements. The building has a prominent entrance with a blue archway. The text "Mangalochkar Institute of Management" is visible on the building's facade. In the foreground, there is a paved area and a small landscaped area with a sculpture.

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5273**

**@gmail.com**

**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of**  
**Management Career Development and**  
**Research.**

**156-B Railway Lines Solapur.**  
**Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject IT in MGT**  
**Programme:- BBAII Sem III**

## Introduction to computer

Computer is derived from Latin word Compute means to calculate

For long Years ago people used stone for calculation like addition, subtraction, multiplication division

They required a machine for calculation

For addition, subtraction, multiplication, division

Abacus was the first machine for calculation

Charles babbage invented a machine was called Analytical engine a Mechanical device

It contains processor, memory

From this machine a computer was developed

So Charles babbage is known as father of computer

### **Defination of computer**

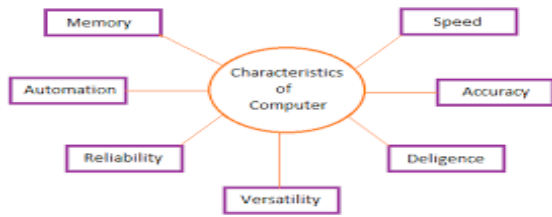
Half Defination

Computer is an electronic device

Full Defination

Computer is an electronic device it accepts data, instruction process data according to instruction and gives out put

Characterstics of computer



## Speed

A computer works with much higher speed and accuracy compared to humans while performing mathematical calculations. Computers can process millions (1,000,000) of instructions per second. The time taken by computers for their operations is microseconds and nanoseconds.

## Accuracy

Computers perform calculations with 100% accuracy. Errors may occur due to data inconsistency or inaccuracy.

## Diligence

A computer can perform millions of tasks or calculations with the same consistency and accuracy. It doesn't feel any fatigue or lack of concentration. Its memory also makes it superior to that of human beings.

## Versatility

Versatility refers to the capability of a computer to perform different kinds of works with same accuracy and efficiency.

## Reliability

A computer is reliable as it gives consistent result for similar set of data i.e., if we give same set of input any number of times, we will get the same result.

### Automation

Computer performs all the tasks automatically i.e. it performs tasks without manual intervention.

## Memory

A computer has built-in memory called primary memory where it stores data. Secondary storage are removable devices such as CDs, pen drives, etc., which are also used to store data.



## Evolution of Hardware and software

### Hardware

Physical parts of computer is called Hardware

### Software

To solve specific problem we require software

Software can be divided in to two categories

1) System software

2)Application softwarwere

### Application software

It is aset of one or more programs designed to solve specific problem or do a specific task

E.g

Tally ,Payroll ,library invoice etc

### System software

It is a set of one or more programs designed to control the operation and extend the processing capability of the computer system

### Functions

- 1) Supports the development of other application software
- 2) Supports the execution of other application softwar
- 3) Communicates with input and output devices
- 4) Monitor the use of hardware devices

Most commonly known types of system software

1)Operating system – Every computer has an operating system which takes care of the effective and efficient utilisation of all the hardware and software

2)Programming language translators

Compiler, Interpreter

3) Communication software

Utility programs

1) Backup

2) Defragmentor

Firmware

In Computing **firmware** is a specific class of computer Software that provides the low-level control for a device's specific hardware. Firmware can either provide a standardized operating environment for more complex device software or, for less complex devices, act as the device's complete Operating system, performing all control, monitoring and data manipulation functions. Typical examples of devices containing firmware are embedded Systems, consumer appliances, computers, computer peripherals, and others. Almost all electronic devices beyond the simplest contain some firmware.

Firmware is held in Non volatile Memory devices such as ROM , EPROM or Flash MEM

### **Applications of computers in various fields**

#### **Following are some examples of uses of computers:**

(1) **Medical Field. Computers** are used in hospital management system, patient history, CT Scan, X-ray, ECG, other medical tests, monitoring and diagnosis purpose.

(2) Homes. ...

(3) Entertainment. ...

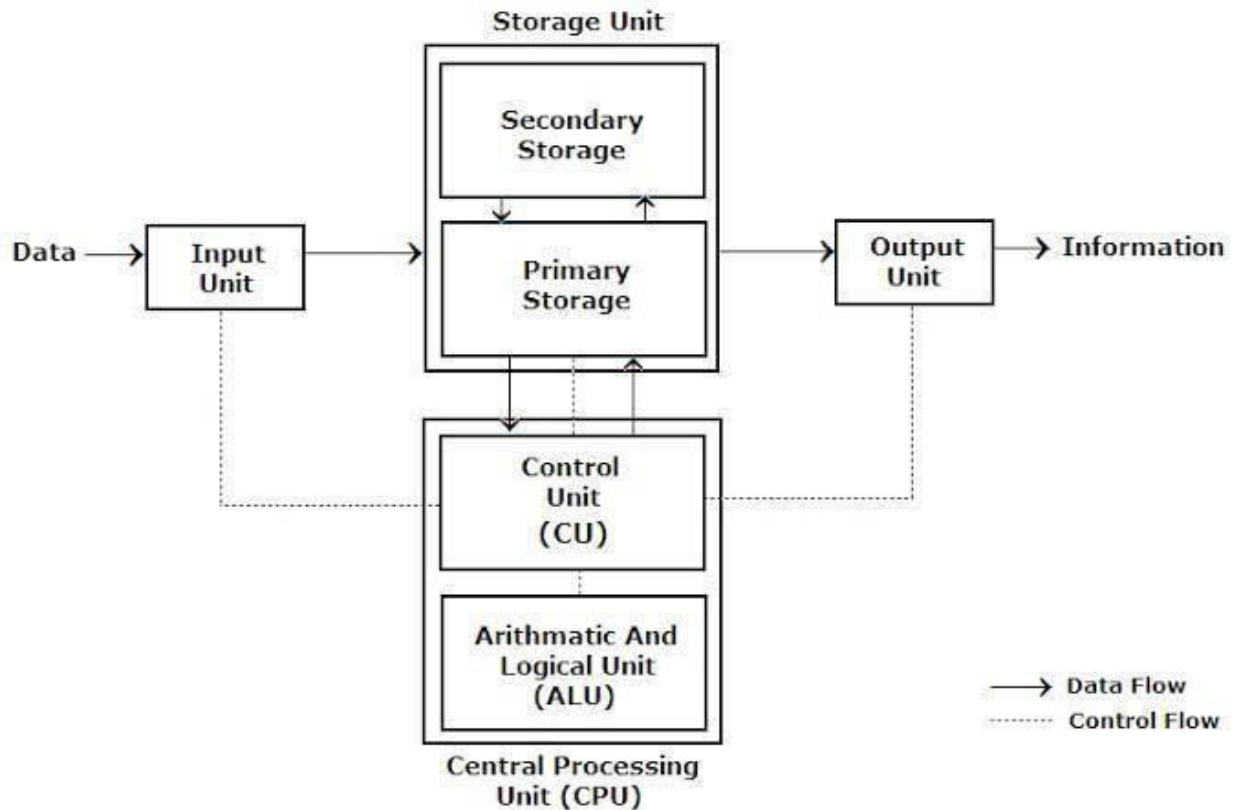
(4) Government. ...

(5) Business. ...

(6) Industry. ...

(7) Education.

Structure of computer:



## Input Unit

This unit contains devices with the help of which we enter data into computer. This unit makes link between user and computer. The input devices translate the information into the form understandable by computer.

## CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results and instructions(program). It controls the operation of all parts of computer.

CPU itself has following three components

- ALU(Arithmetic Logic Unit)
- Memory Unit
- Control Unit

### Output Unit

Output unit consists of devices with the help of which we get the information from computer. This unit is a link between computer and users. Output devices translate the computer's output into the form understandable by users.

Sr.No.	Operation	Description
1	Take Input	The process of entering data and instructions into the computer system
2	Store Data	Saving data and instructions so that they are available for processing as and when required.
3	Processing Data	Performing arithmetic, and logical operations on data in order to convert them into useful information.
4	Output Information	The process of producing useful information or results for the user, such as a printed report or visual display.
5	Control the workflow	Directs the manner and sequence in which all of the above operations are performed.

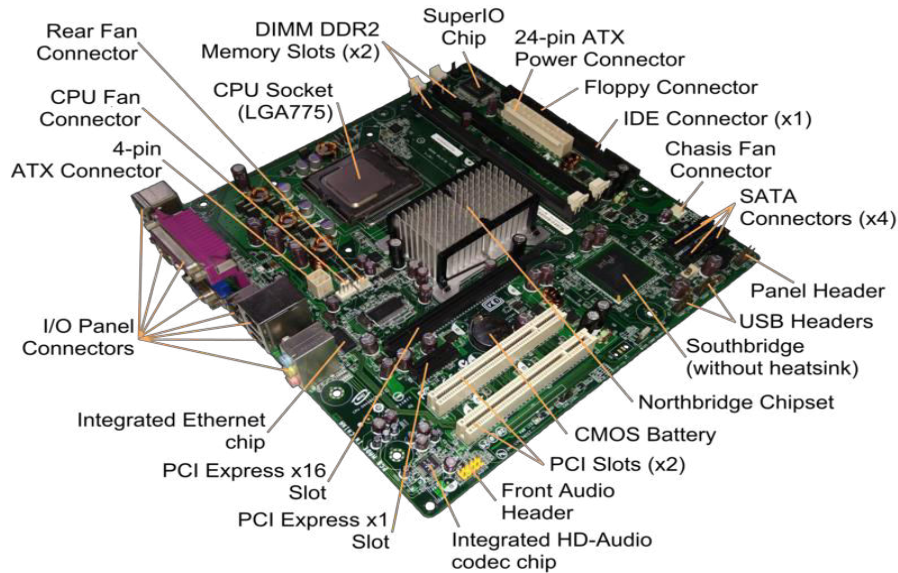
# Input Devices of Computer



# Output Devices



# Introduction to Motherboard



- A **motherboard** (also called **mainboard**, **main circuit board**, **system board**, **baseboard**, **planar board**, **logic board**, <sup>[1]</sup> and **mobo**) is the main printed\_circuit board (PCB) in general-purpose computers and other expandable systems. It holds, and allows communication between, many of the crucial electronic components of a system, such as the central processing unit (CPU) and memory, and provides connectors for other peripherals. Unlike a backplane, a motherboard usually contains significant sub-systems, such as the central processor, the chipset's input/output and memory controllers, interface connectors, and other components integrated for general use.
- *Motherboard* means specifically a PCB with expansion capabilities. As the name suggests, this board is often referred to as the "mother" of all components attached to it, which often include peripherals, interface cards, and : [sound](#)

[cards](#), video cards, network cards, hard drives, and other forms of persistent storage; TV tuner cards, cards providing extra USB or FireWire slots; and a variety of other custom components.

- Similarly, the term *mainboard* describes a device with a single board and no additional expansions or capability, such as controlling boards in laser printers, television sets, washing machines, mobile phones, and other embedded systems with limited expansion abilities.

SMPS (Switched Mode Power Supply



SMPS stands for switch-**mode** power supply. Its job is to convert wall-voltage AC power to lower voltage DC power. Most computer chips in modern computers require power in the general neighborhood of 1.2-3.3V, with some older devices requiring between 5-12V DC

SMPS stands for switch-**mode** power supply. Its job is to convert wall-voltage AC power to lower voltage DC power. Most computer chips in modern computers require power in the general neighborhood of 1.2-3.3V, with some older devices requiring between 5-12V DC

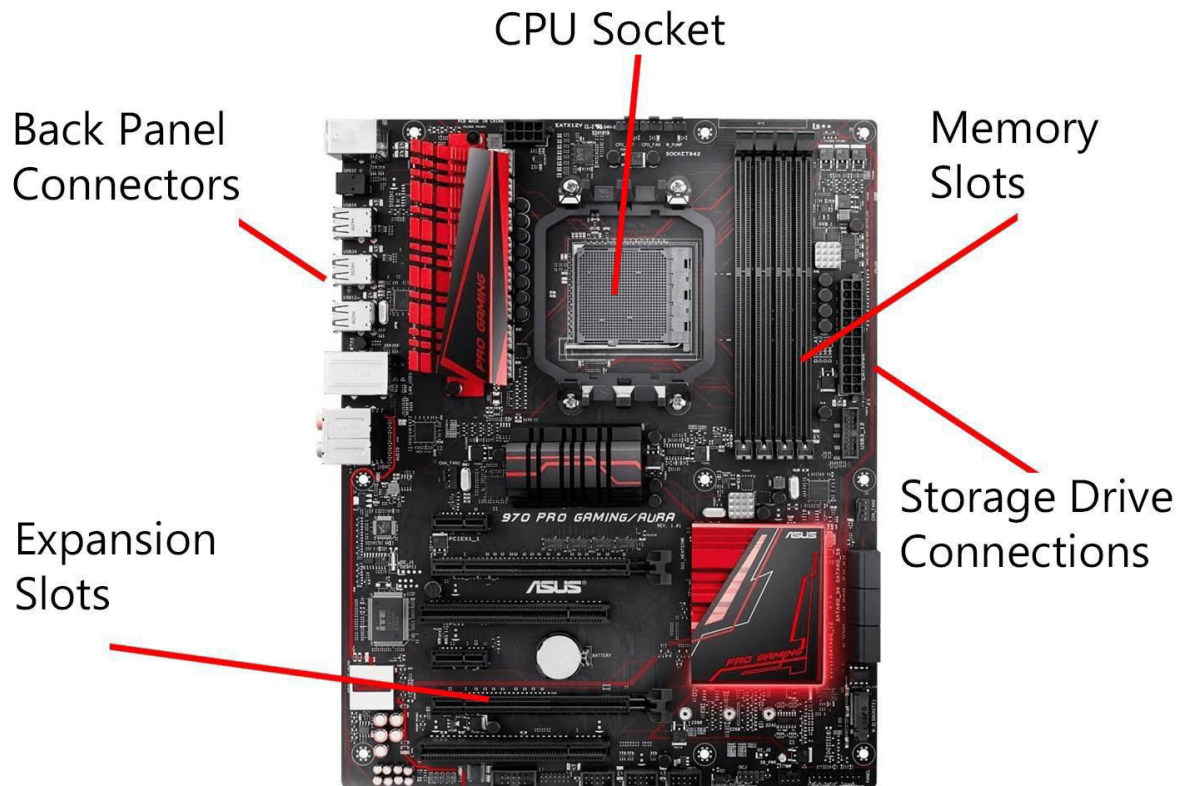
Math Coprocessor



- The **math coprocessor** was an optional add-on for the Intel 8086, 80386 and 80486 processors that allowed computers to perform faster **mathematical** calculations, increasing its overall performance. Today, all computer processors are released with a **math coprocessor** incorporated onto the processor.



## Expansion slots



- Alternatively known as a **bus slot** or **expansion port**, an **expansion slot** is a connection or port inside a computer on the motherboard or riser card. It provides an installation point for a hardware expansion card to be connected. For example, if you wanted to install a new video card in the computer, you'd purchase a video expansion card and install that card into the compatible expansion slot.
- Below is a listing of expansion slots commonly found in a computer and the devices associated with those slots. Clicking on any of the links below provide you with additional details.
- **AGP** - Video card.
- **AMR** - Modem, sound card.
- **CNR** - Modem, network card, sound card.

- **EISA** - SCSI, network card, video card.
- **ISA** - Network card, sound card, video card.
- **PCI** - Network card, SCSI, sound card, video card.
- **PCI Express** - Video card, modem, sound card, network card.
- **VESA** - Video card.

#### Serial and parallel ports



- The main difference between a **serial port** and a **parallel port** is that a **serial port** transmits data one bit after another, while a **parallel port** transmits all 8 bits of a byte in **parallel**. ... Computers have both **serial and parallel ports** along with newer technology called a USB (Universal **Serial Bus**) **port**

## Limitation of Computer

- **Computer** cannot operate without the instructions given by humans. It is programmed to work effectively, fast and accurately. **Computer** cannot think by itself and does not have common sense.

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh52**

**73**

**@gmail.com**

**Faculty-Nilima Mondhe**

**Subject-C++ II**

**Sem- IV**

In C++, inheritance is a process in which one object gets all the data members and Functions of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called **derived class** and the class whose members are inherited is called **base class**. The derived class is the specialized class for the base class.

---

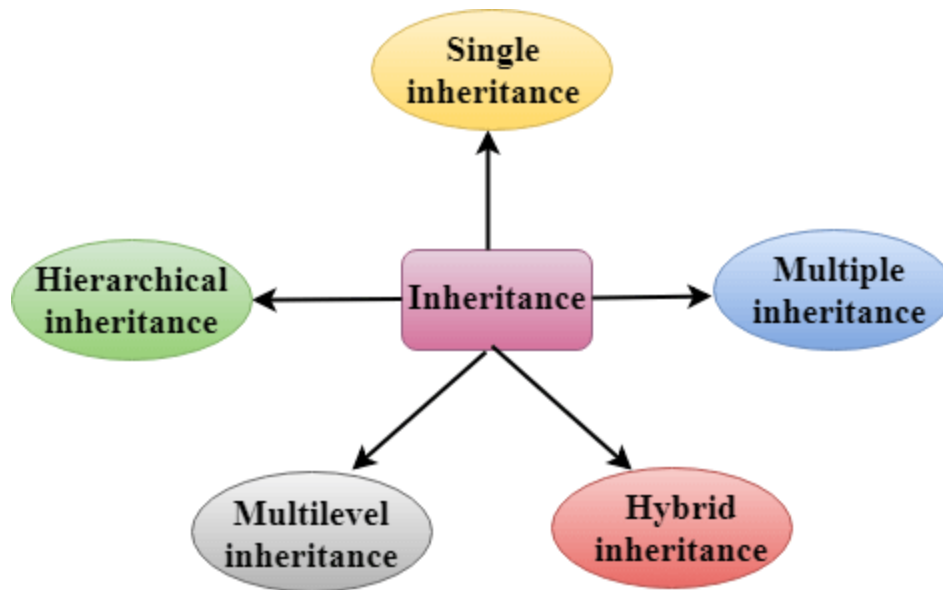
## Advantage of C++ Inheritance

**Code reusability:** Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

## Types Of Inheritance

**C++ supports five types of inheritance:**

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance



## Derived Classes

A Derived class is defined as the class derived from the base class.

The Syntax of Derived class:

1. **class** derived\_class\_name :: visibility-mode base\_class\_name
2. {
3.   // body of the derived class.
4. }

**Where,**

**derived\_class\_name:** It is the name of the derived class.

**visibility mode:** The visibility mode specifies whether the features of the base class are publicly inherited or privately inherited. It can be public or private.

**base\_class\_name:** It is the name of the base class.

- When the base class is privately inherited by the derived class, public members of the base class becomes the private members of the derived class. Therefore, the public members of the base class are not accessible by the objects of the derived class only by the member functions of the derived class.

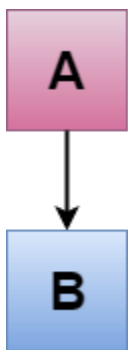
- When the base class is publicly inherited by the derived class, public members of the base class also become the public members of the derived class. Therefore, the public members of the base class are accessible by the objects of the derived class as well as by the member functions of the base class.

**Note:**

- In C++, the default mode of visibility is private.
- The private members of the base class are never inherited.

## C++ Single Inheritance

**Single inheritance** is defined as the inheritance in which a derived class is inherited from the only one base class.



Where 'A' is the base class, and 'B' is the derived class.

## C++ Single Level Inheritance Example: Inheriting Fields

When one class inherits another class, it is known as single level inheritance. Let's see the example of single level inheritance which inherits the fields only.

1. `#include <iostream>`
2. `using namespace std;`
3. `class Account {`
4. `public:`
5. `float salary = 60000;`
6. `};`
7. `class Programmer: public Account {`

```
8.  public:
9.  float bonus = 5000;
10. };
11. int main(void) {
12.     Programmer p1;
13.     cout<<"Salary: "<<p1.salary<<endl;
14.     cout<<"Bonus: "<<p1.bonus<<endl;
15.     return 0;
16. }
```

Output:

```
Salary: 60000
Bonus: 5000
```

In the above example, Employee is the **base** class and Programmer is the **derived** class.

## C++ Single Level Inheritance Example: Inheriting Methods

Let's see another example of inheritance in C++ which inherits methods only.

```
1. #include <iostream>
2. using namespace std;
3. class Animal {
4.     public:
5.     void eat() {
6.         cout<<"Eating..."<<endl;
7.     }
8. };
9. class Dog: public Animal
10. {
11.     public:
12.     void bark(){
13.         cout<<"Barking...";
14.     }
```



```
15. };
16. int main(void) {
17.     Dog d1;
18.     d1.eat();
19.     d1.bark();
20.     return 0;
21. }
```

Output:

```
Eating...
Barking...
```

Let's see a simple example.

```
1. #include <iostream>
2. using namespace std;
3. class A
4. {
5.     int a = 4;
6.     int b = 5;
7.     public:
8.     int mul()
9.     {
10.         int c = a*b;
11.         return c;
12.     }
13. };
14.
15. class B : private A
16. {
17.     public:
18.     void display()
19.     {
20.         int result = mul();
21.         std::cout << "Multiplication of a and b is : " << result << std::endl;
```

```
22. }
23. };
24. int main()
25. {
26.     B b;
27.     b.display();
28.
29.     return 0;
30. }
```

Output:

```
Multiplication of a and b is : 20
```

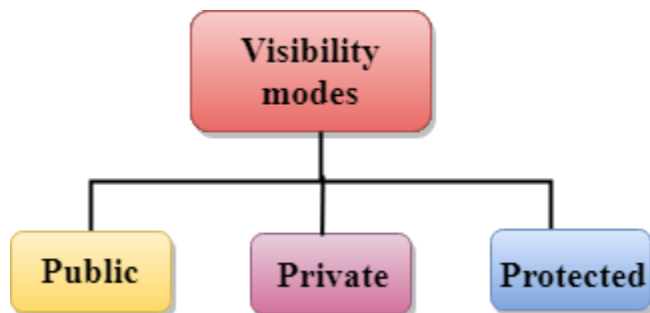
In the above example, class A is privately inherited. Therefore, the mul() function of class 'A' cannot be accessed by the object of class B. It can only be accessed by the member function of class B.

## How to make a Private Member Inheritable

The private member is not inheritable. If we modify the visibility mode by making it public, but this takes away the advantage of data hiding.

C++ introduces a third visibility modifier, i.e., **protected**. The member which is declared as protected will be accessible to all the member functions within the class as well as the class immediately derived from it.

**Visibility modes can be classified into three categories:**



- **Public:** When the member is declared as public, it is accessible to all the functions of the program.

- **Private:** When the member is declared as private, it is accessible within the class only.
- **Protected:** When the member is declared as protected, it is accessible within its own class as well as the class immediately derived from it.

## Visibility of Inherited Members

Base class visibility	Derived class visibility		
	Public	Private	Protected
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

## C++ Multilevel Inheritance

**Multilevel inheritance** is a process of deriving a class from another derived class.



## C++ Multi Level Inheritance Example

When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

Let's see the example of multi level inheritance in C++.

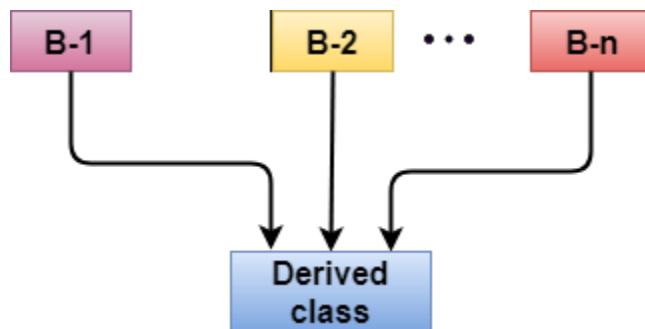
```
1. #include <iostream>
2. using namespace std;
3. class Animal {
4.     public:
5.     void eat() {
6.         cout<<"Eating..."<<endl;
7.     }
8. };
9. class Dog: public Animal
10. {
11.     public:
12.     void bark(){
13.         cout<<"Barking..."<<endl;
14.     }
15. };
16. class BabyDog: public Dog
17. {
18.     public:
19.     void weep() {
20.         cout<<"Weeping...";
21.     }
22. };
23. int main(void) {
24.     BabyDog d1;
25.     d1.eat();
26.     d1.bark();
27.     d1.weep();
28.     return 0;
29. }
```

Output:

```
Eating...  
Barking...  
Weeping...
```

## C++ Multiple Inheritance

**Multiple inheritance** is the process of deriving a new class that inherits the attributes from two or more classes.



**Syntax of the Derived class:**

1. **class** D : visibility B-1, visibility B-2, ?
2. {
3. // Body of the class;
4. }

Let's see a simple example of multiple inheritance.

1. **#include** <iostream>
2. **using namespace** std;
3. **class** A
4. {
5. **protected:**
6. **int** a;
7. **public:**
8. **void** get\_a(**int** n)
9. {
10. a = n;

```
11. }
12. };
13.
14. class B
15. {
16.     protected:
17.     int b;
18.     public:
19.     void get_b(int n)
20.     {
21.         b = n;
22.     }
23. };
24. class C : public A, public B
25. {
26.     public:
27.     void display()
28.     {
29.         std::cout << "The value of a is : " <<a<< std::endl;
30.         std::cout << "The value of b is : " <<b<< std::endl;
31.         cout<<"Addition of a and b is : "<<a+b;
32.     }
33. };
34. int main()
35. {
36.     C c;
37.     c.get_a(10);
38.     c.get_b(20);
39.     c.display();
40.
41.     return 0;
42. }
```

Output:

```
The value of a is : 10
The value of b is : 20
Addition of a and b is : 30
```

In the above example, class 'C' inherits two base classes 'A' and 'B' in a public mode.

## Ambiguity Resolution in Inheritance

Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base class.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class A
4. {
5.     public:
6.     void display()
7.     {
8.         std::cout << "Class A" << std::endl;
9.     }
10.};
11. class B
12. {
13.     public:
14.     void display()
15.     {
16.         std::cout << "Class B" << std::endl;
17.     }
18.};
19. class C : public A, public B
20. {
21.     void view()
22.     {
23.         display();
24.     }
```

```
25.};
26. int main()
27. {
28.   C c;
29.   c.display();
30.   return 0;
31. }
```

Output:

```
error: reference to 'display' is ambiguous
      display();
```

- The above issue can be resolved by using the class resolution operator with the function. In the above example, the derived class code can be rewritten as:

```
1. class C : public A, public B
2. {
3.   void view()
4.   {
5.     A :: display();    // Calling the display() function of class A.
6.     B :: display();    // Calling the display() function of class B.
7.
8.   }
9. };
```

An ambiguity can also occur in single inheritance.

Consider the following situation:

```
1. class A
2. {
3.   public:
4.   void display()
5.   {
6.     cout << "?Class A?";
7.   }
8. };
```



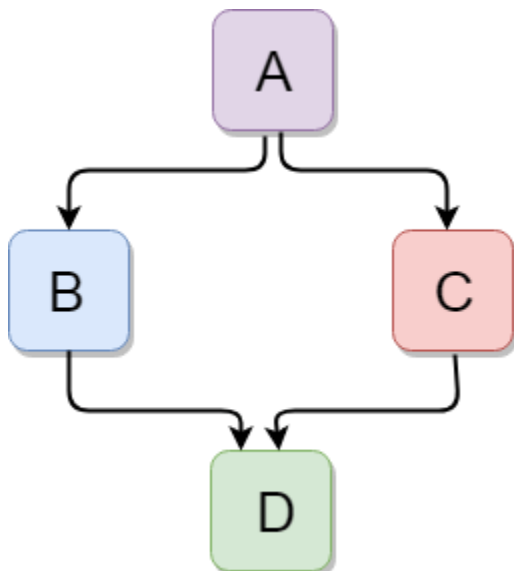
```
9. class B
10. {
11. public:
12. void display()
13. {
14. cout<<"Class B?";
15. }
16. };
```

In the above case, the function of the derived class overrides the method of the base class. Therefore, call to the display() function will simply call the function defined in the derived class. If we want to invoke the base class function, we can use the class resolution operator.

```
1. int main()
2. {
3.   B b;
4.   b.display();           // Calling the display() function of B class.
5.   b.B :: display();     // Calling the display() function defined in B class.
6. }
```

## C++ Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance.



Let's see a simple example:

```
1. #include <iostream>
2. using namespace std;
3. class A
4. {
5.     protected:
6.     int a;
7.     public:
8.     void get_a()
9.     {
10.         std::cout << "Enter the value of 'a' : " << std::endl;
11.         cin >> a;
12.     }
13. };
14.
15. class B : public A
16. {
17.     protected:
18.     int b;
19.     public:
20.     void get_b()
21.     {
22.         std::cout << "Enter the value of 'b' : " << std::endl;
23.         cin >> b;
24.     }
25. };
26. class C
27. {
28.     protected:
29.     int c;
30.     public:
31.     void get_c()
32.     {
```

```

33.     std::cout << "Enter the value of c is : " << std::endl;
34.     cin>>c;
35. }
36. };
37.
38. class D : public B, public C
39. {
40.     protected:
41.     int d;
42.     public:
43.     void mul()
44.     {
45.         get_a();
46.         get_b();
47.         get_c();
48.         std::cout << "Multiplication of a,b,c is : " <<a*b*c<< std::endl;
49.     }
50. };
51. int main()
52. {
53.     D d;
54.     d.mul();
55.     return 0;
56. }

```

Output:

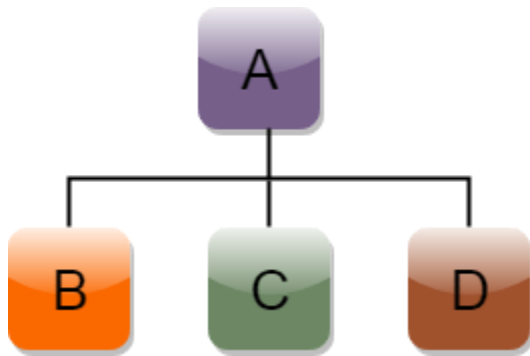
```

Enter the value of 'a' :
10
Enter the value of 'b' :
20
Enter the value of c is :
30
Multiplication of a,b,c is : 6000

```

## C++ Hierarchical Inheritance

Hierarchical inheritance is defined as the process of deriving more than one class from a base class.



**Syntax of Hierarchical inheritance:**

```
1. class A
2. {
3.     // body of the class A.
4. }
5. class B : public A
6. {
7.     // body of class B.
8. }
9. class C : public A
10. {
11.     // body of class C.
12. }
13. class D : public A
14. {
15.     // body of class D.
16. }
```

Let's see a simple example:

```
1. #include <iostream>
2. using namespace std;
3. class Shape           // Declaration of base class.
4. {
```

```
5.  public:
6.  int a;
7.  int b;
8.  void get_data(int n,int m)
9.  {
10.     a= n;
11.     b = m;
12. }
13.};
14. class Rectangle : public Shape // inheriting Shape class
15. {
16.  public:
17.  int rect_area()
18.  {
19.     int result = a*b;
20.     return result;
21. }
22.};
23. class Triangle : public Shape // inheriting Shape class
24. {
25.  public:
26.  int triangle_area()
27.  {
28.     float result = 0.5*a*b;
29.     return result;
30. }
31.};
32. int main()
33. {
34.  Rectangle r;
35.  Triangle t;
36.  int length,breadth,base,height;
37.  std::cout << "Enter the length and breadth of a rectangle: " << std::endl;
38.  cin>>length>>breadth;
```

```

39. r.get_data(length,breadth);
40. int m = r.rect_area();
41. std::cout << "Area of the rectangle is : " <<m<< std::endl;
42. std::cout << "Enter the base and height of the triangle: " << std::endl;
43. cin>>base>>height;
44. t.get_data(base,height);
45. float n = t.triangle_area();
46. std::cout <<"Area of the triangle is : " << n<<std::endl;
47. return 0;
48.}

```

The word “polymorphism” means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism is a person who at the same time can have different characteristics. Like a man at the same time is a father, a husband and an employee. So the same person exhibits different behavior in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object-Oriented Programming. **In C++, polymorphism is mainly divided into two types:**

- Compile-time Polymorphism
- Runtime Polymorphism

#### *Types of Polymorphism*

1. **Compile-time polymorphism:** This type of polymorphism is achieved by function overloading or operator overloading.
  - **Function Overloading:** When there are multiple functions with the same name but different parameters, then the functions are said to be **overloaded**. Functions can be overloaded by **changing the number of arguments** or/and **changing the type of arguments**. [Rules of Function Overloading](#)

- CPP

```
// C++ program for function overloading
```

```
#include <bits/stdc++.h>

using namespace std;

class Geeks
{
    public:

    // function with 1 int parameter
    void func(int x)
    {
        cout << "value of x is " << x << endl;
    }

    // function with same name but 1 double parameter
    void func(double x)
    {
        cout << "value of x is " << x << endl;
    }

    // function with same name and 2 int parameters
```

```
void func(int x, int y)

{

    cout << "value of x and y is " << x << ", " << y << endl;

}

};

int main() {

    Geeks obj1;

    // Which function is called will depend on the parameters passed

    // The first 'func' is called

    obj1.func(7);

    // The second 'func' is called

    obj1.func(9.132);

    // The third 'func' is called

    obj1.func(85,64);

    return 0;

}
```



```
}
```

- **Output:**

value of x is 7

value of x is 9.132

value of x and y is 85, 64

- In the above example, a single function named *func* acts differently in three different situations, which is a property of polymorphism.
- **Operator Overloading**: C++ also provides the option to overload operators. For example, we can make use of the addition operator (+) for string class to concatenate two strings. We know that the task of this operator is to add two operands. So a single operator '+', when placed between integer operands, adds them and when placed between string operands, concatenates them. **Example:**

- CPP

```
// CPP program to illustrate
// Operator Overloading

#include<iostream>

using namespace std;

class Complex {

private:

    int real, imag;

public:

    Complex(int r = 0, int i =0) {real = r;   imag = i;}
```

```

// This is automatically called when '+' is used with
// between two Complex objects

Complex operator + (Complex const &obj) {

    Complex res;

    res.real = real + obj.real;

    res.imag = imag + obj.imag;

    return res;

}

void print() { cout << real << " + i" << imag << endl; }

};

int main()

{

    Complex c1(10, 5), c2(2, 4);

    Complex c3 = c1 + c2; // An example call to "operator+"

    c3.print();

}

```

- **Output:**

**12 + i9**

- In the above example, the operator '+' is overloaded. Usually, this operator is used to add two numbers (integers or floating point numbers), but here the operator is made to perform the addition of two imaginary or complex numbers. To learn about operator overloading in detail, visit [this](#) link.

1. **Runtime polymorphism**: This type of polymorphism is achieved by Function Overriding.
  - **Function overriding** occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

- CPP

```
// C++ program for function overriding

#include <bits/stdc++.h>

using namespace std;

class base
{
public:

    virtual void print ()

    { cout<< "print base class" <<endl; }

    void show ()

    { cout<< "show base class" <<endl; }

};

class derived:public base
```

```
{

public:

    void print () //print () is already virtual function in derived class, we
could also declared as virtual void print () explicitly

    { cout<< "print derived class" <<endl; }

    void show ()

    { cout<< "show derived class" <<endl; }

};

//main function

int main()

{

    base *bptr;

    derived d;

    bptr = &d;

    //virtual function, binded at runtime (Runtime polymorphism)

    bptr->print();

    // Non-virtual function, binded at compile time
```

```
bptr->show();  
  
return 0;  
}
```

- Output:  
print derived class  
show base class



**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of Management**  
**Career Development and Research.**

**156-B Railway Lines Solapur.**

**Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject Cyber Law**

**Programme:- BCAII Sem IV**

# what is Cybercrime

- Cyber crime or computer-oriented crime is a crime that includes a computer and a network. The computer may have been used in the execution of a crime or it may be the target.  
Cyber crime is the use of a computer as a weapon for committing crimes such as committing fraud, identities theft or breaching privacy. Cyber crime, especially through the Internet, has grown in importance as the computer has become central to every field like commerce, entertainment and government. Cyber crime may endanger a person or a nation's security and financial health.
- Cyber crime encloses a wide range of activities but these can generally be divided into two categories:
- Crimes that aim computer networks or devices. These types of crimes involves different threats (like virus, bugs etc.) and denial-of-service (DoS) attacks.
- Crimes that use computer networks to commit other criminal activities. These types of crimes include cyber stalking, financial fraud or identity theft.

# Types of Cyber Crime:

- **Cyber Terrorism:**

Cyber terrorism is the use of the computer and internet to perform violent acts that result in loss of life. This may include different type of activities either by software or hardware for threatening life of citizens.

In general, Cyber terrorism can be defined as an act of terrorism committed through the use of cyberspace or computer resources.

- **Cyber Extortion:**

Cyber extortion occurs when a website, e-mail server or computer system is subjected to or threatened with repeated denial of service or other attacks by malicious hackers. These hackers demand huge money in return for assurance to stop the attacks and to offer protection.



- **Cyber Warfare:**

Cyber warfare is the use or targeting in a battle space or warfare context of computers, online control systems and networks. It involves both offensive and defensive operations concerning to the threat of cyber attacks, espionage and sabotage.

- **Internet Fraud:**

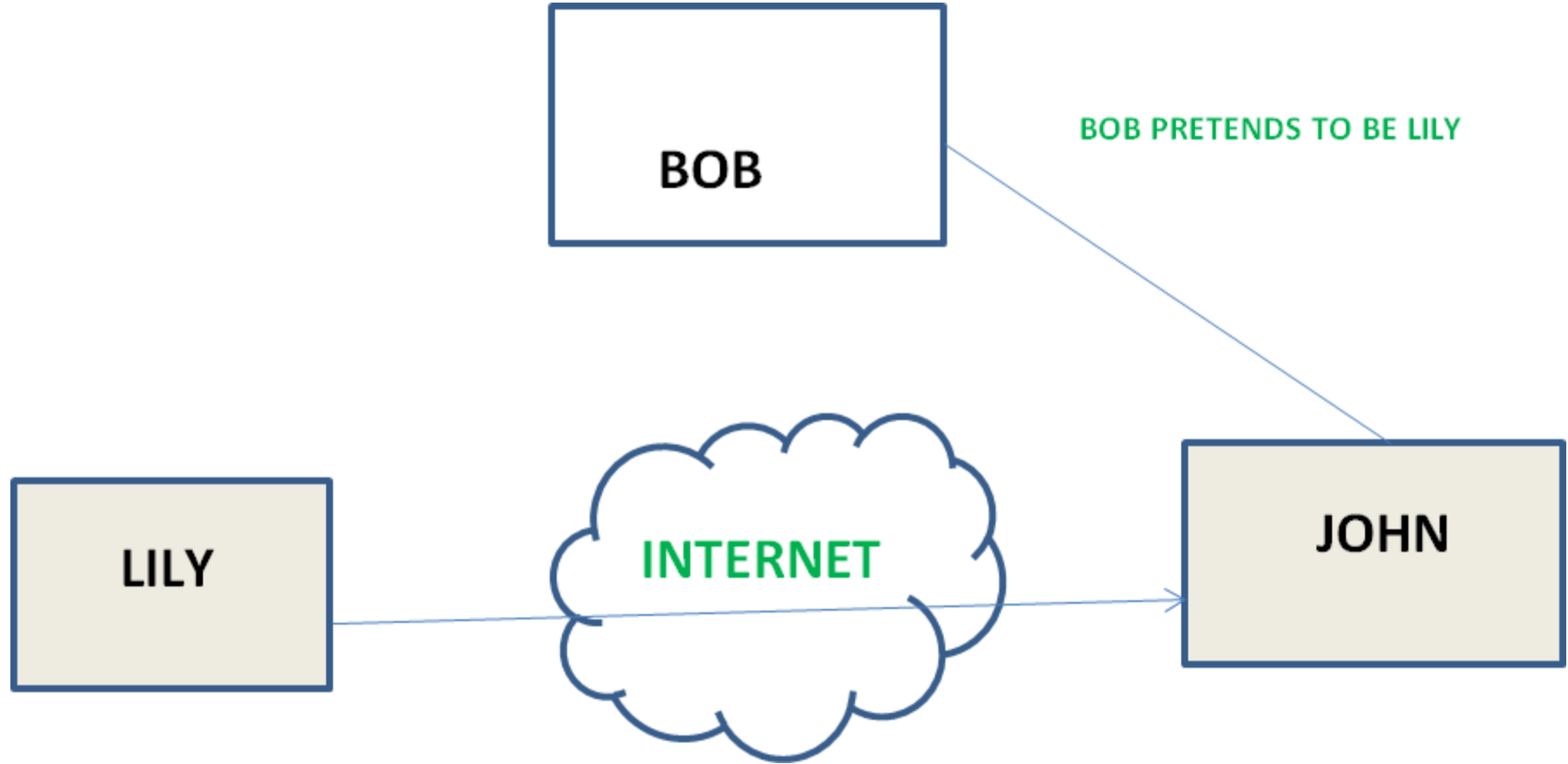
Internet fraud is a type of fraud or deceit which makes use of the Internet and could include hiding of information or providing incorrect information for the purpose of deceiving victims for money or property. Internet fraud is not considered a single, distinctive crime but covers a range of illegal and illicit actions that are committed in cyberspace.

- **Cyber Stalking:**

This is a kind of online harassment wherein the victim is subjected to a barrage of online messages and emails. In this case, these stalkers know their victims and instead of offline stalking, they use the Internet to stalk. However, if they notice that cyber stalking is not having the desired effect, they begin offline stalking along with cyber stalking to make the victims' lives more miserable.

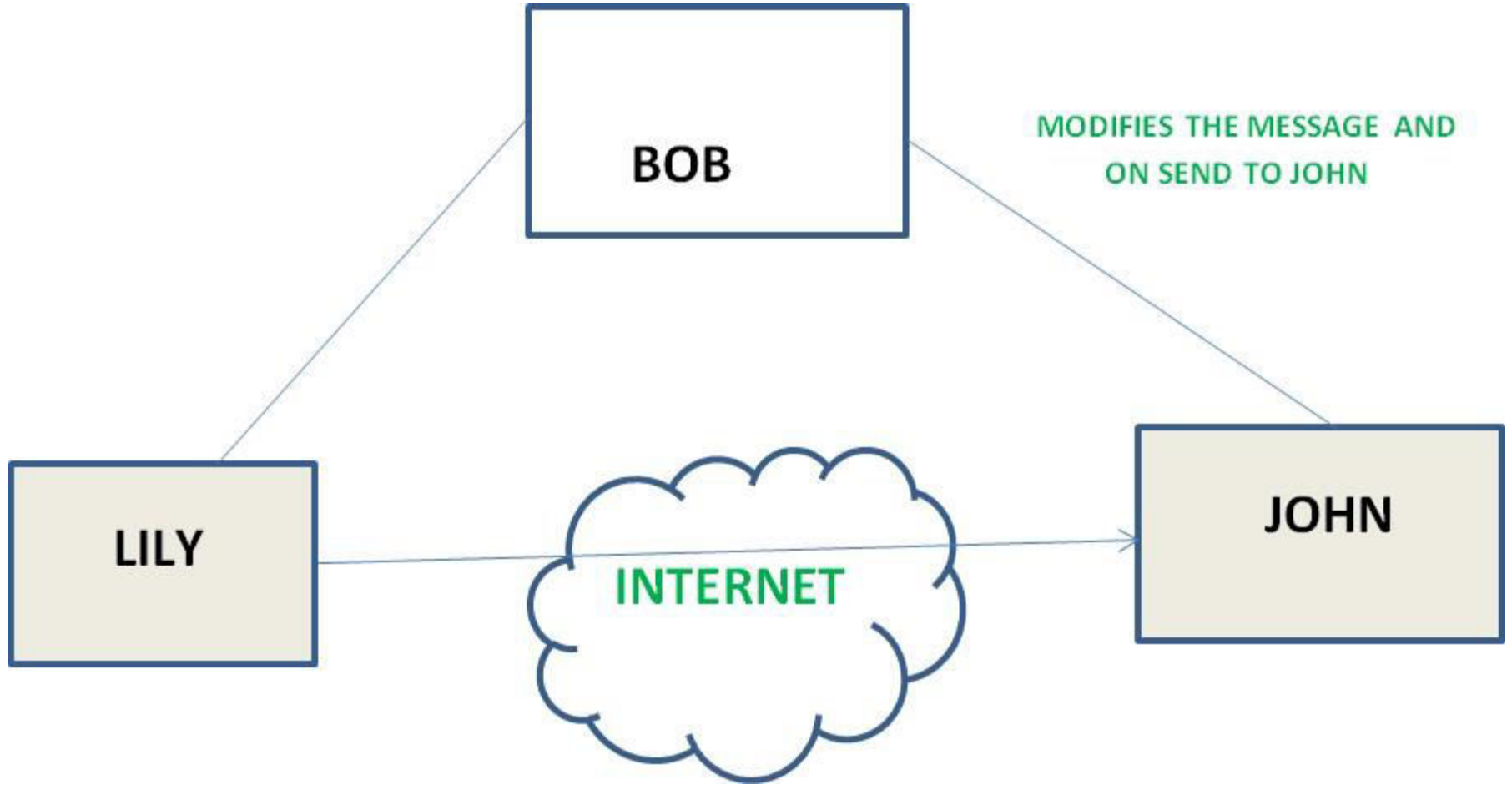
# Classifications of Security attacks (Passive Attacks and Active Attacks),

- **Active attacks:** An Active attack attempts to alter system resources or effect their operations. Active attack involve some modification of the data stream or creation of false statement. Types of active attacks are as following:
- **Masquerade –**  
Masquerade attack takes place when one entity pretends to be different entity. A Masquerade attack involves one of the other form of active attacks.



- **Modification of messages –**

It means that some portion of a message is altered or that message is delayed or reordered to produce an unauthorised effect. For example, a message meaning “Allow JOHN to read confidential file X” is modified as “Allow Smith to read confidential file X”.



- **Repudiation –**

This attack is done by either sender or receiver. The sender or receiver can deny later that he/she has send or receive a message. For example, customer ask his Bank “To transfer an amount to someone” and later on the sender(customer) deny that he had made such a request. This is repudiation.

- **Replay –**

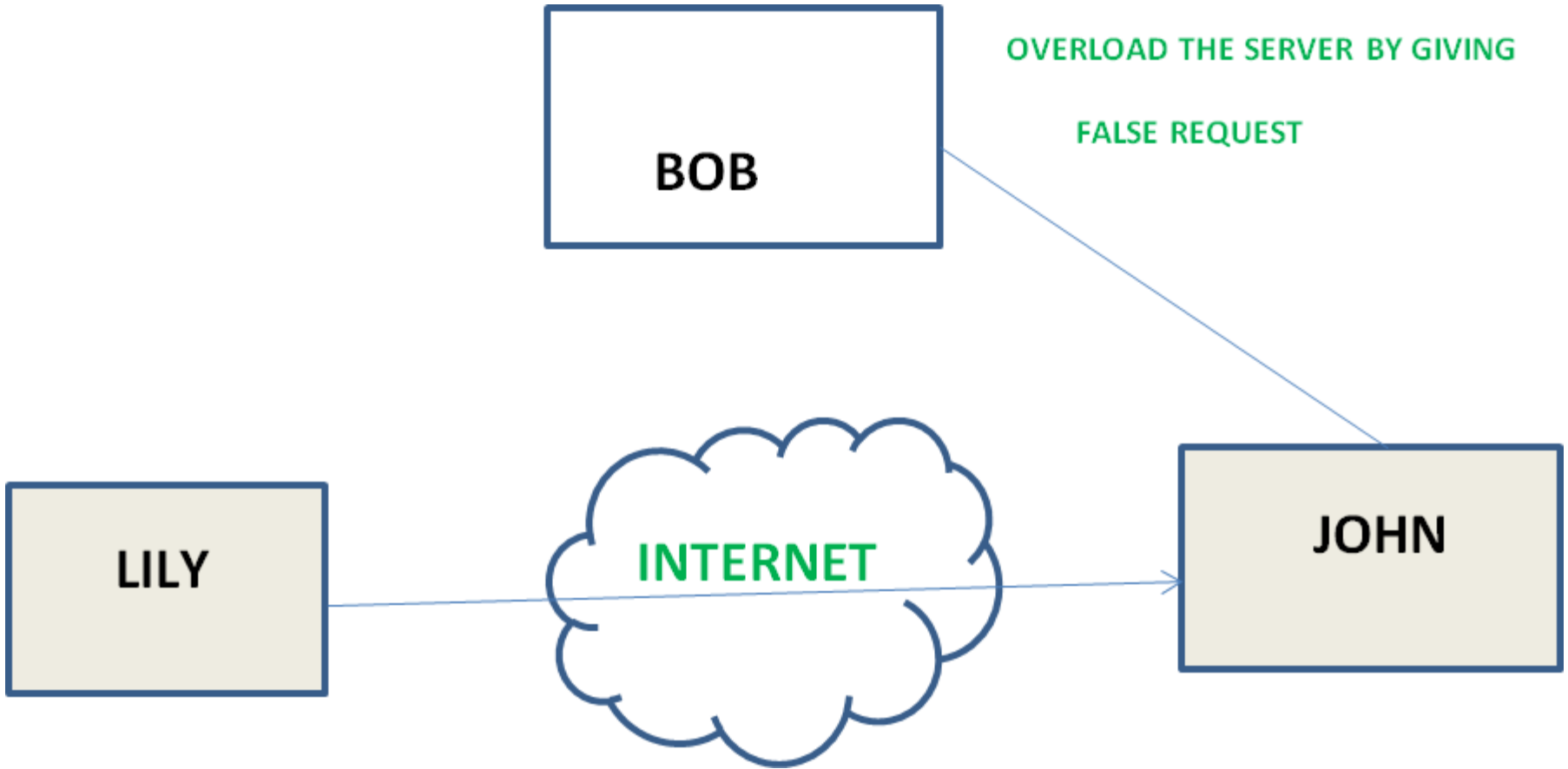
It involves the passive capture of a message and its subsequent the transmission to produce an authorized effect.



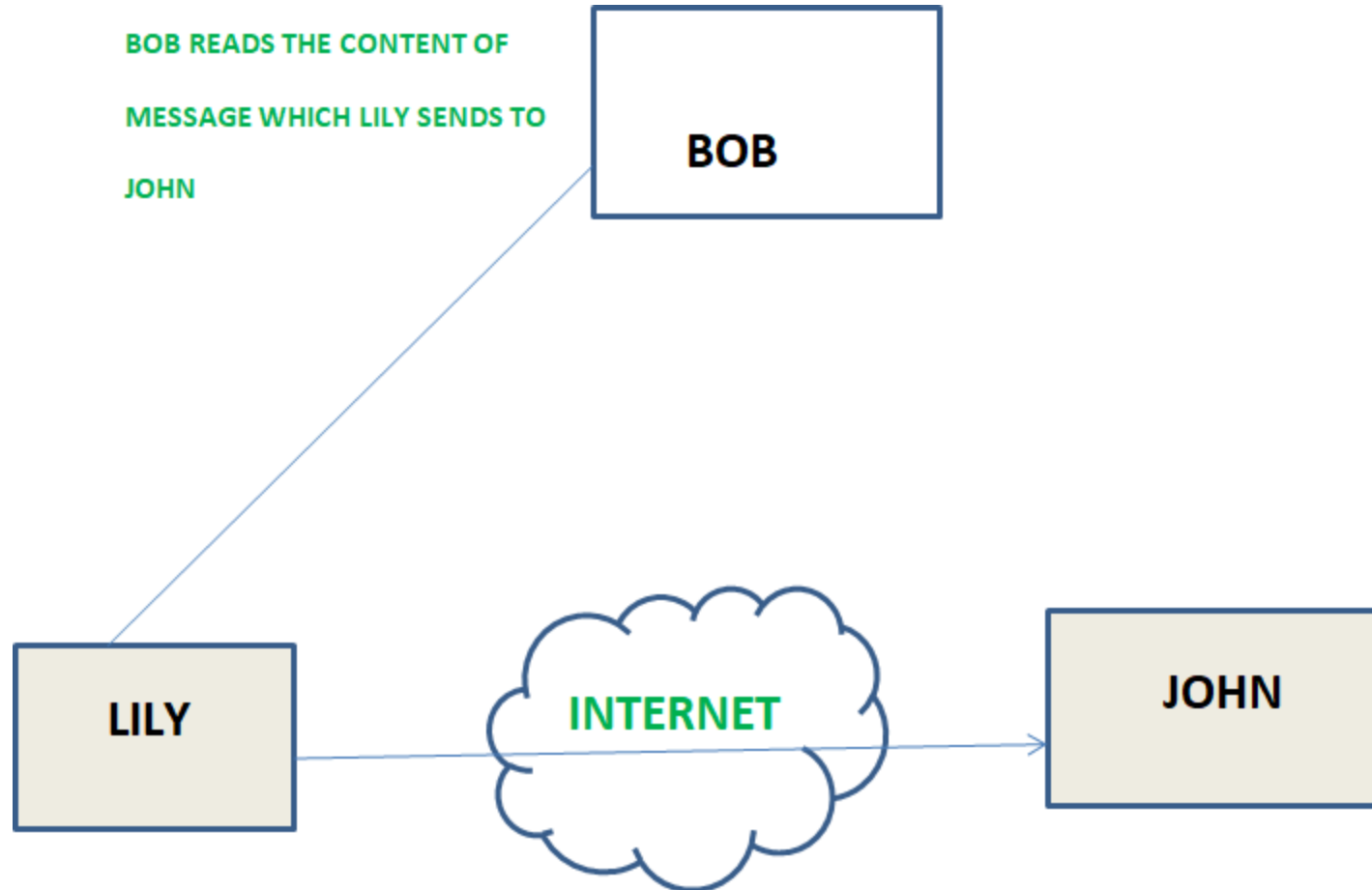


- **Denial of Service –**

It prevents normal use of communication facilities. This attack may have a specific target. For example, an entity may suppress all messages directed to a particular destination. Another form of service denial is the disruption of an entire network either by disabling the network or by overloading it by messages so as to degrade performance.



- **Passive attacks:** A Passive attack attempts to learn or make use of information from the system but does not affect system resources. Passive Attacks are in the nature of eavesdropping on or monitoring of transmission. The goal of the opponent is to obtain information is being transmitted. Types of Passive attacks are as following:
- **The release of message content –**  
Telephonic conversation, an electronic mail message or a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

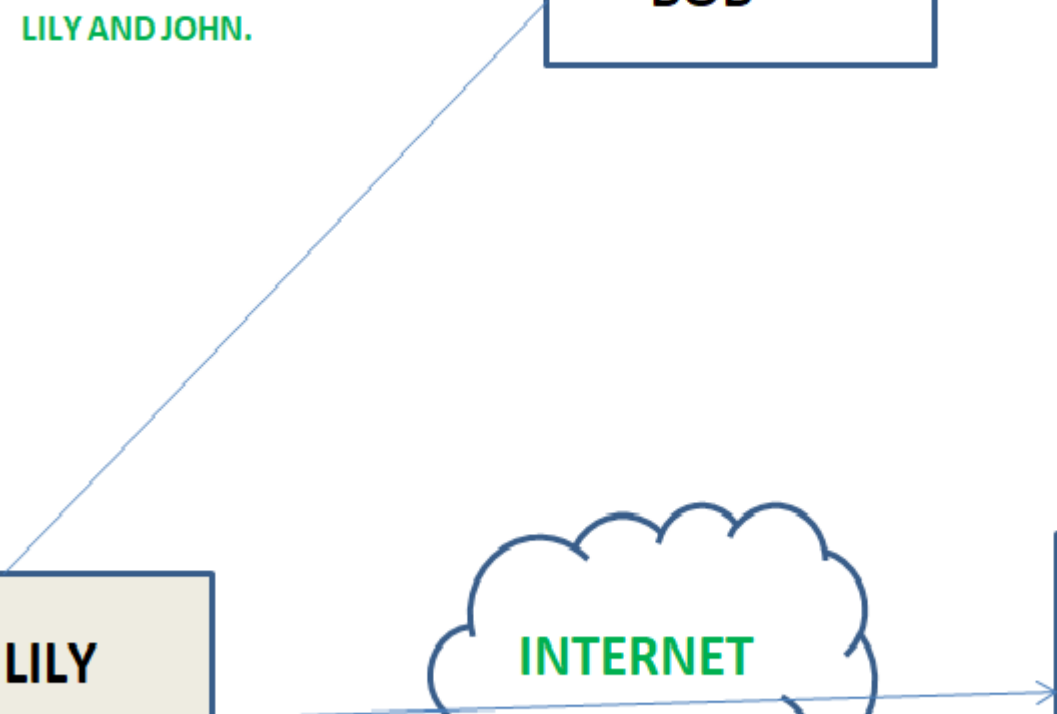
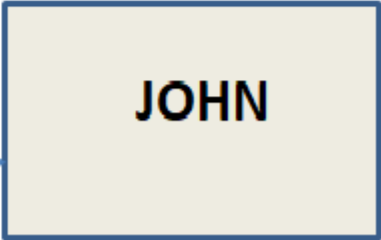
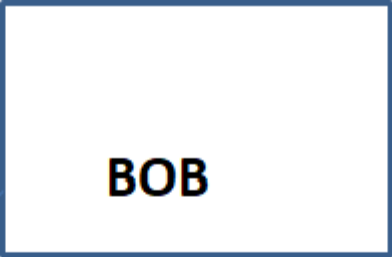


- **Traffic analysis –**

Suppose that we had a way of masking (encryption) of information, so that the attacker even if captured the message could not extract any information from the message.

The opponent could determine the location and identity of communicating host and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

**BOB OBSERVES THE PATTERN OF  
MESSAGES EXCHANGED BETWEEN  
LILY AND JOHN.**



# Essential Terminology

- **Threat** can be anything that can take advantage of a vulnerability to breach security and negatively alter, erase, harm object or objects of interest.
- **Software attacks** means attack by Viruses, Worms, Trojan Horses etc. Many users believe that malware, virus, worms, bots are all same things. But they are not same, only similarity is that they all are malicious software that behave differently.

- **Malware** is a combination of 2 terms- Malicious and Software. So Malware basically means malicious software that can be an intrusive program code or a anything that is designed to perform malicious operations on system. Malware can be divided in 2 categories:
  - Infection Methods
  - Malware Actions



- **Virus** – They have the ability to replicate themselves by hooking them to the program on the host computer like songs, videos etc and then they travel all over the Internet. The Creeper Virus was first detected on ARPANET. Examples include File Virus, Macro Virus, Boot Sector Virus, Stealth Virus etc.
- **Worms** – Worms are also self replicating in nature but they don't hook themselves to the program on host computer. Biggest difference between virus and worms is that worms are network aware. They can easily travel from one computer to another if network is available and on the target machine they will not do much harm, they will for example consume hard disk space thus slowing down the computer.

- Malware on the **basis of Actions:**
- **Adware** – Adware is not exactly malicious but they do breach privacy of the users. They display ads on computer's desktop or inside individual programs. They come attached with free to use software, thus main source of revenue for such developers. They monitor your interests and display relevant ads. An attacker can embed malicious code inside the software and adware can monitor your system activities and can even compromise your machine.
- **Spyware** – It is a program or we can say a software that monitors your activities on computer and reveal collected information to interested party. Spyware are generally dropped by Trojans, viruses or worms. Once dropped they installs themselves and sits silently to avoid detection.

- **Theft of intellectual property** means violation of intellectual property rights like copyrights, patents etc.
- **Identity theft** means to act someone else to obtain person's personal information or to access vital information they have like accessing the computer or social media account of a person by login into the account by using their login credentials.

- **Theft of equipment and information** is increasing these days due to the mobile nature of devices and increasing information capacity.
- **Social media attacks** – In this cyber criminals identify and infect a cluster of websites that persons of a particular organisation visit, to steal information.
- **Mobile Malware** –There is a saying when there is a connectivity to Internet there will be danger to Security. Same goes to Mobile phones where gaming applications are designed to lure customer to download the game and unintentionally they will install malware or virus in the device.

- **Corporate data on personal devices** – These days every organization follows a rule BYOD. BYOD means Bring your own device like Laptops, Tablets to the workplace. Clearly BYOD pose a serious threat to security of data but due to productivity issues organizations are arguing to adopt this.

# Vulnerabilities

- **Vulnerabilities** are weaknesses in a system that gives threats the opportunity to compromise assets. All systems have vulnerabilities. Even though the technologies are improving but the number of vulnerabilities are increasing such as tens of millions of lines of code, many developers, human weaknesses, etc. Vulnerabilities mostly happened because of Hardware, Software, Network and Procedural vulnerabilities.

- **. Software Vulnerability:**

A software error happen in development or configuration such as the execution of it can violate the security policy. For examples:

- Lack of input validation
- Unverified uploads
- Cross-site scripting
- Unencrypted data, etc.

- **Network Vulnerability:**

A weakness happen in network which can be hardware or software.

For examples:

- Unprotected communication
- Malware or malicious software (e.g.:Viruses, Keyloggers, Worms, etc)
- Social engineering attacks
- Misconfigured firewalls



- **Procedural Vulnerability:**

A weakness happen in an organization operational methods.

For examples:

- Password procedure – Password should follow the standard password policy.
- Training procedure – Employees must know which actions should be taken and what to do to handle the security. Employees must never be asked for user credentials online. Make the employees know social engineering and phishing threats.



**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5273**

**@gmail.com**



**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of Management**  
**Career Development and Research.**

156-B Railway Lines Solapur.

S.P. Mandali Pune

Name of Faculty: **Mr. Santosh Kulkarni**  
Affiliated P.H. Solapur University

Subject DBMS

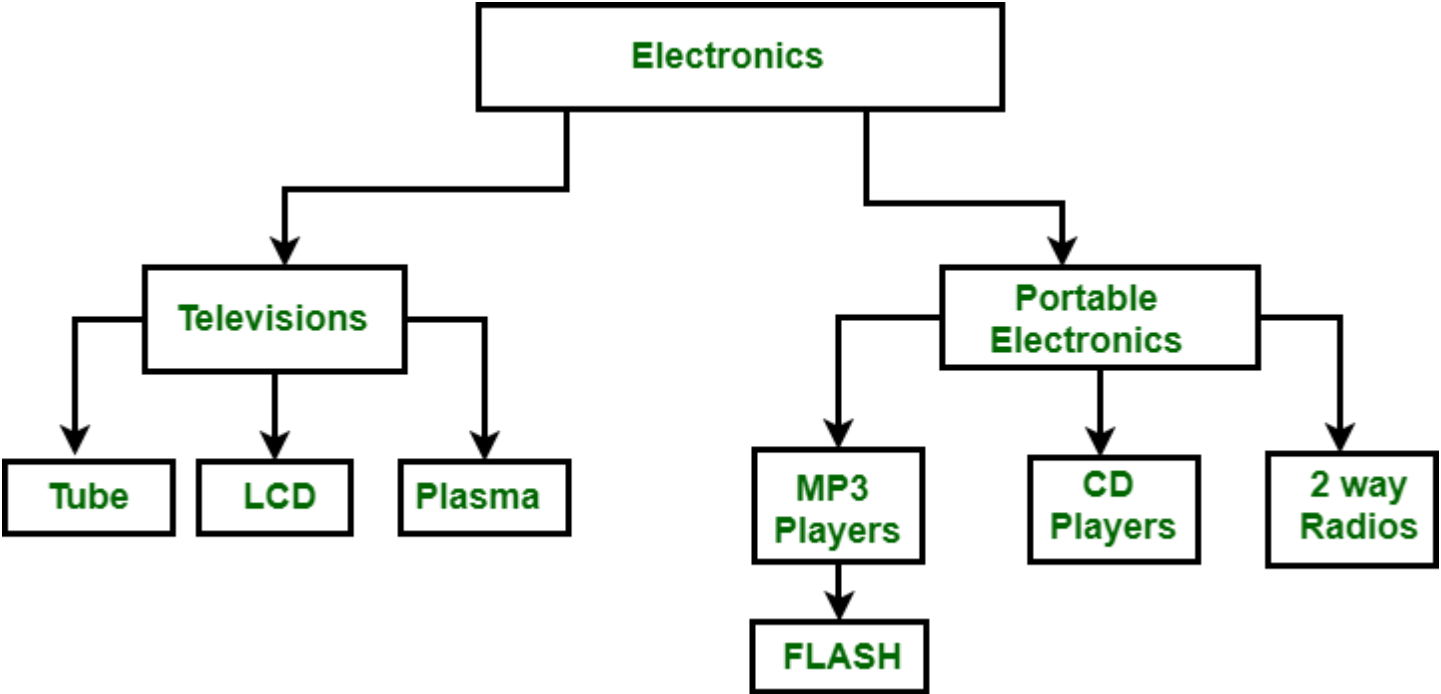
Programme:- BCAII Sem IV

# Database Design

- Types of data models- Relational, Network, Hierarchical
- Hierarchical Data Model



- Hierarchical data model is the oldest type of the data model. It was developed by IBM in 1968. It organizes data in the tree-like structure. Hierarchical model consists of the the following :
  - It contains nodes which are connected by branches.
  - The topmost node is called the root node.
  - If there are multiple nodes appear at the top level, then these can be called as root segments.
  - Each node has exactly one parent.
  - One parent may have many child.

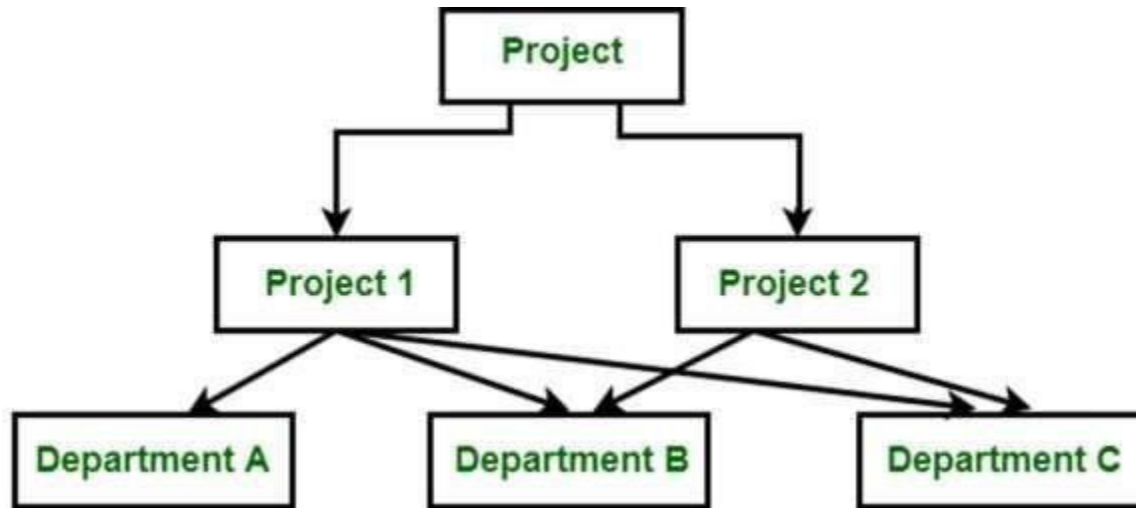




- In the above figure, Electronics is the root node which has two children i.e. Televisions and Portable Electronics. These two has further children for which they act as parent. For example: Television has children as Tube, LCD and Plasma, for these three Television act as parent. It follows one to many relationship.

## 2. Network Data Model :

- It is the advance version of the hierarchical data model. To organize data it uses directed graphs instead of the tree-structure. In this child can have more than one parent. It uses the concept of the two data structures i.e. Records and Sets.



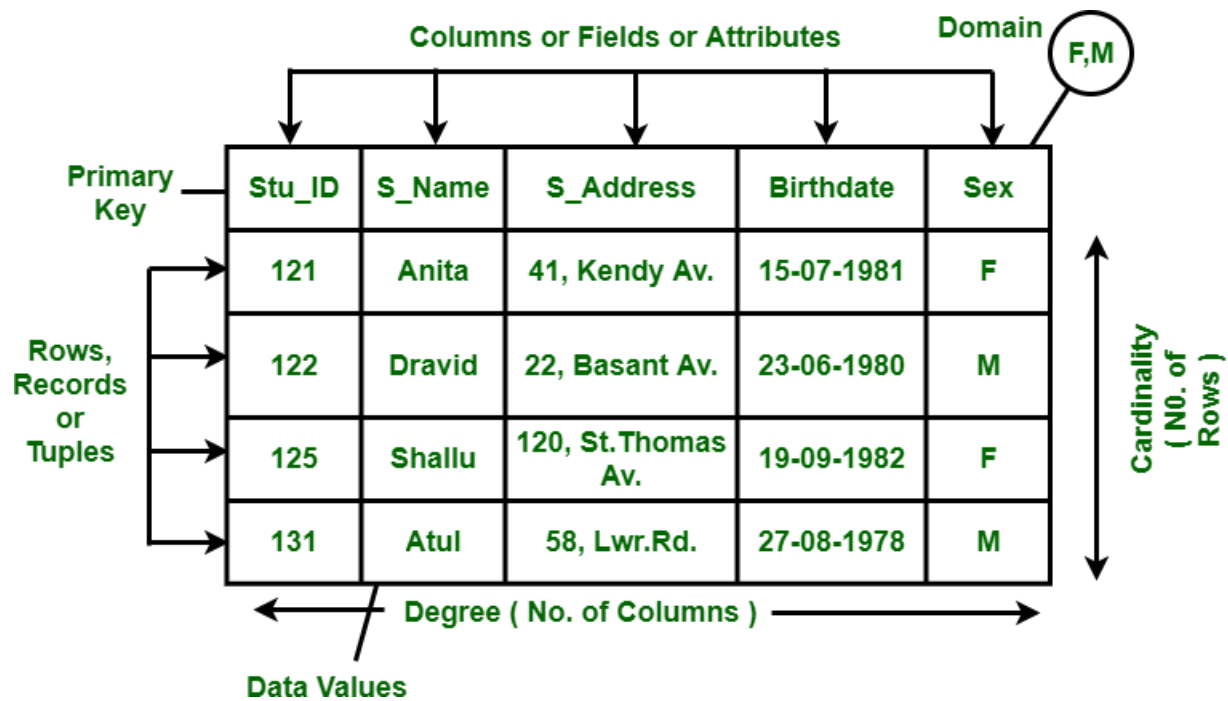
- In the above figure, Project is the root node which has two children i.e. Project 1 and Project 2. Project 1 has 3 children and Project 2 has 2 children. Total there are 5 children i.e. Department A, Department B and Department C, they are network related children as we said that this model can have more than one parent. So, for the Department B and Department C have two parents i.e. Project 1 and Project 2.

- In the above diagram , Project is the root node which has two children i.e. Project 1 and Project 2. Project 1 has 3 children and Project 2 has 2 children. Total there are 5 children i.e Department A, Department B and Department C, they are network related children as we said that this model can have more than one parent. So, for the Department B and Department C have two parents i.e. Project 1 and Project 2.

# Relational Model

- Relational Model was proposed by E.F. Codd to model data in the form of relations or tables. After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages like Oracle SQL, MySQL etc. So we will see what Relational Model is.
- **What is Relational Model?**
- Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL\_NO, NAME, ADDRESS, PHONE and AGE shown in Table 1.

<b>ROLL_NO</b>	<b>NAME</b>	<b>ADDRESS</b>	<b>PHONE</b>	<b>AGE</b>
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		1





- The relational data model was introduced by C. F. Codd in 1970. Currently, it is the most widely used data model. The relational data model describes the world as “a collection of inter-related relations (or tables).” A relational data model involves the use of data tables that collect groups of elements into relations. These models work based on the idea that each table setup will include a primary key or identifier. Other tables use that identifier to provide "relational" data links and results.
- Today, there are many commercial Relational Database Management System (RDBMS), such as Oracle, IBM DB2, and Microsoft SQL Server. There are also many free and open-source RDBMS, such as MySQL, mSQL (mini-SQL) and the embedded Java DB (Apache Derby). Database administrators use Structured Query Language (SQL) to retrieve data elements from a relational database.
-

- As mentioned, the primary key is a fundamental tool in creating and using relational data models. It must be unique for each member of a data set. It must be populated for all members. Inconsistencies can cause problems in how developers retrieve data. Other issues with relational database designs include excessive duplication of data, faulty or partial data, or improper links or associations between tables. A large part of routine database administration involves evaluating all the data sets in a database to make sure that they are consistently populated and will respond well to SQL or any other data retrieval method.
- For example, a conventional database row would represent a tuple, which is a set of data that revolves around an instance or virtual object so that the primary key is its unique identifier. A column name in a data table is associated with an attribute, an identifier or feature that all parts of a data set have. These and other strict conventions help to provide database administrators and designers with standards for crafting relational database setups.



## FOREIGN KEY



### STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNT RY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

### STUDENT\_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

<u>studentId</u>	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

Foreign Keys

Relationship

Primary Keys

<u>courseId</u>	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

- E-R model: entities, attributes and its types, Relationship, Relationship
- **Entity, Entity Type, Entity Set –**
- An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.
- An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



Entity Type



Entity Set

- **Attribute(s):**

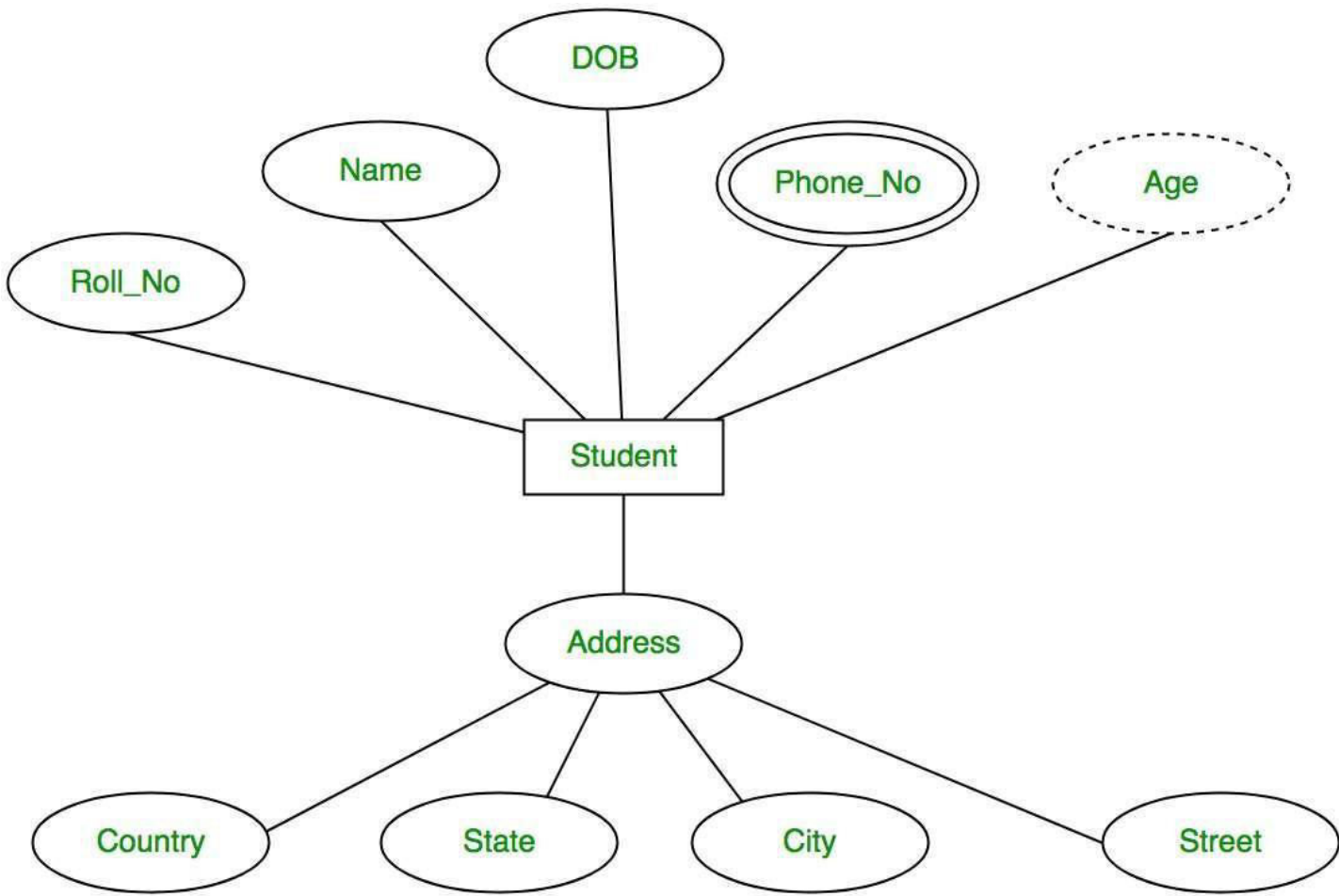
Attributes are the **properties which define the entity type**. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

-

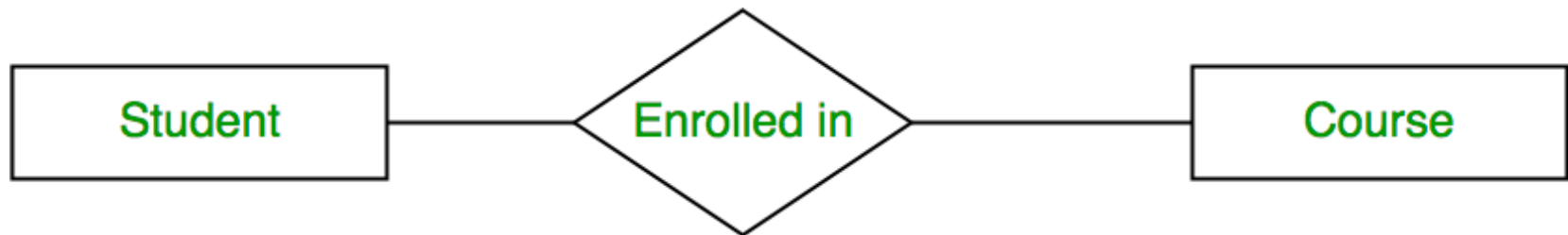




Attribute



- **Relationship Type and Relationship Set:**



- Attributes
- Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.
- There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

# Types of Attributes

- **Simple attribute** – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute** – Composite attributes are made of more than one simple attribute. For example, a student's complete name may have `first_name` and `last_name`.
- **Derived attribute** – Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, `average_salary` in a department should not be saved directly in the database, instead it can be derived. For another example, `age` can be derived from `data_of_birth`.

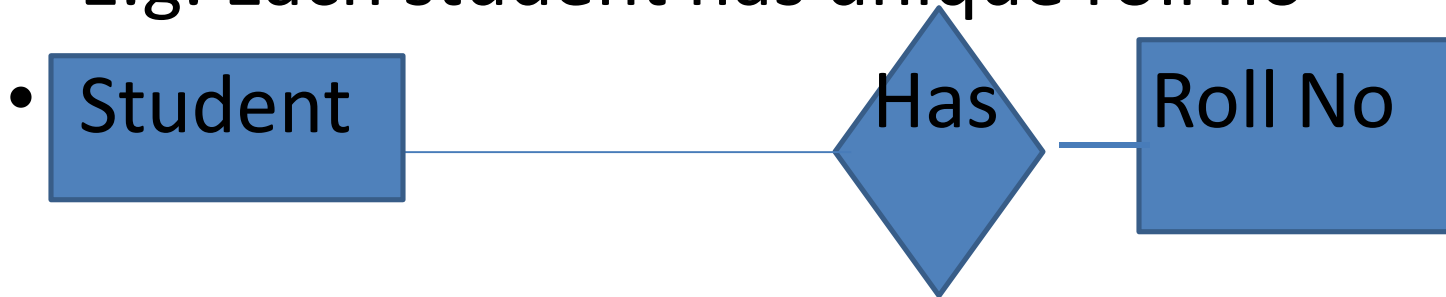
- **Single-value attribute** – Single-value attributes contain single value. For example – Social\_Security\_Number.
- **Multi-value attribute** – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

- Relationship
- The association among entities is called a relationship. For example, an employee **works\_at** a department, a student **enrolls** in a course. Here, Works\_at and Enrolls are called relationships.

- Relationship Set
- A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.



- **One-to-one** – One entity from entity set A can be associated with at most one entity of entity set B and vice versa.
- E.g. Each student has unique roll no



- **Many to one** – When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is

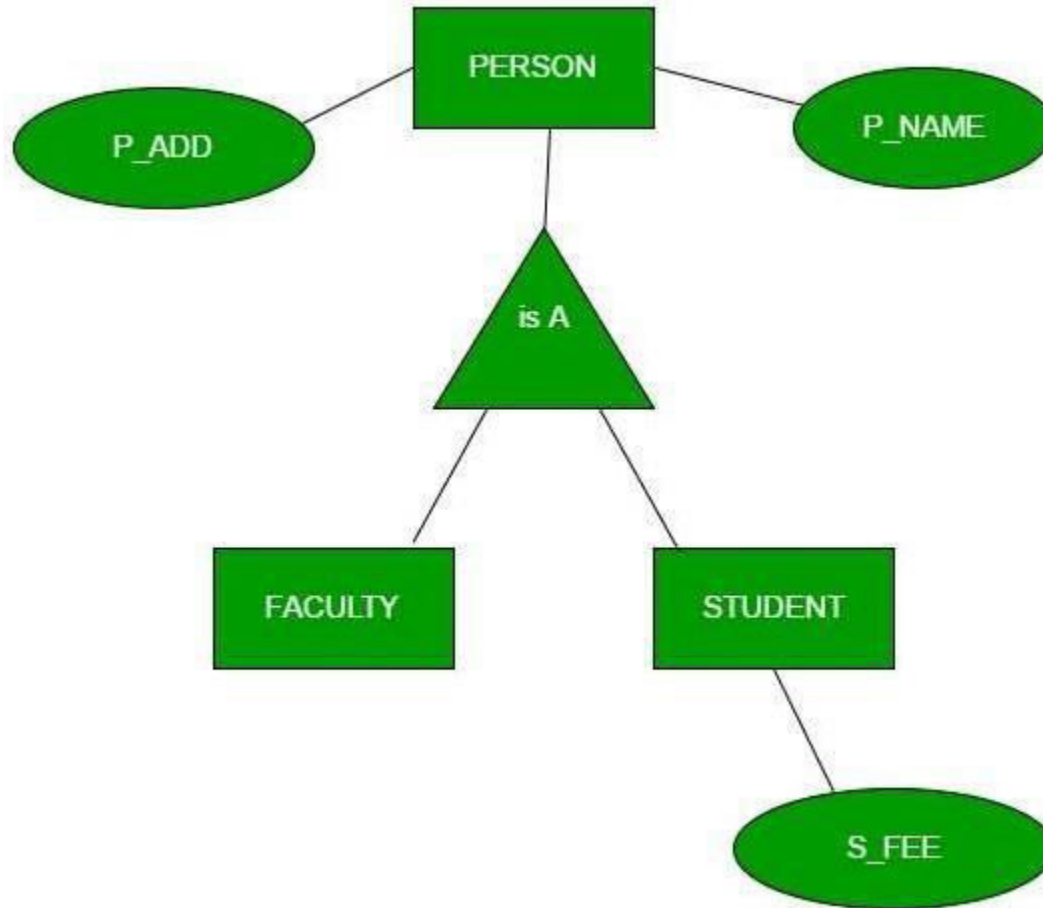


- **Many to many** – When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



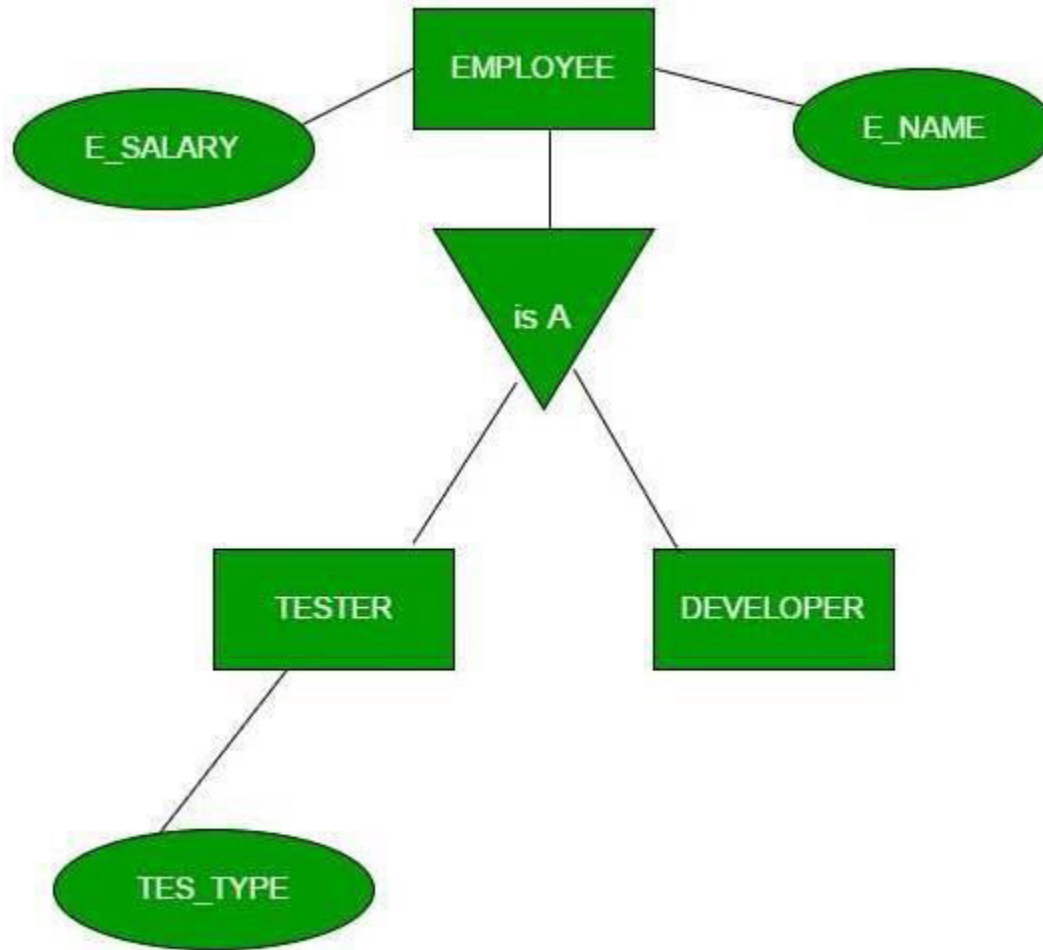
# Generalization

- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P\_NAME, P\_ADD become part of higher entity (PERSON) and specialized attributes like S\_FEE become part of specialized entity (STUDENT).



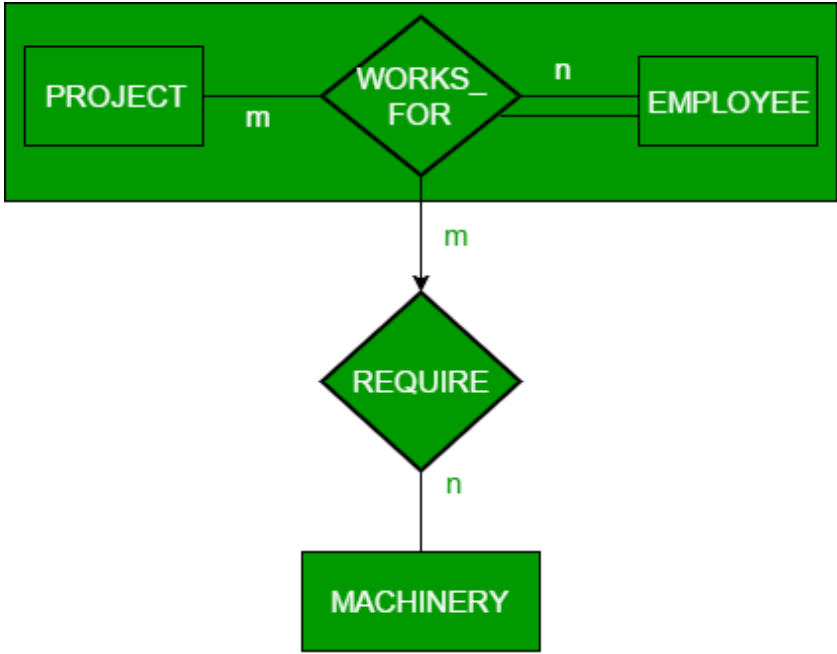
- **Specialization –**
- n specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E\_NAME, E\_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES\_TYPE become part of specialized entity (TESTER).





Specialization

- **Aggregation –**
- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS\_FOR and entity MACHINERY. Using aggregation, WORKS\_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



Aggregation

- Relational Model: Relation, Domain, Tuples, Degree, cardinality
- Relational Model was proposed by E.F. Codd to model data in the form of relations or tables. After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages like Oracle SQL, MySQL etc. So we will see what Relational Model is.

- What is Relational Model?
- Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL\_NO, NAME, ADDRESS, PHONE and AGE shown in Table 1.

<b>ROLL_NO</b>	<b>NAME</b>	<b>ADDRESS</b>	<b>PHONE</b>	<b>AGE</b>
	<b>STUDENT</b>			
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

- **Attribute:** Attributes are the properties that define a relation. e.g.; **ROLL\_NO, NAME**
- **Tuple:** Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as:

---

RAM

DELHI

9455123451

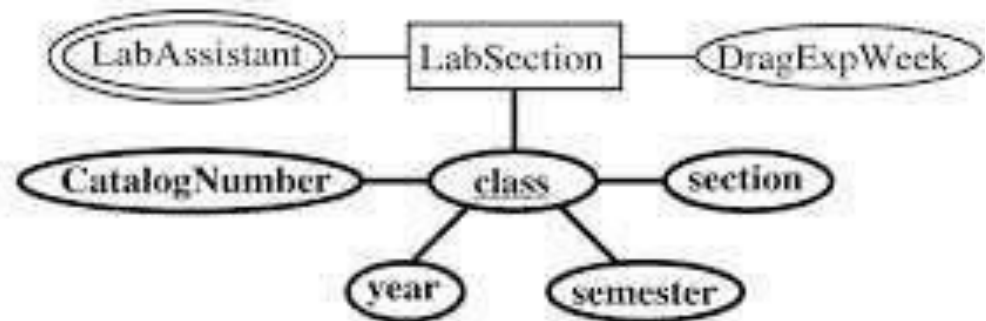
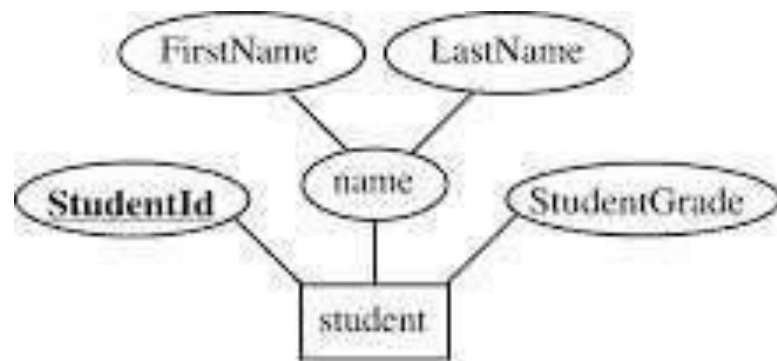
18



- **Degree:** The number of attributes in the relation is known as degree of the relation. The **STUDENT** relation defined above has degree 5.

-

- **Cardinality:** The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
- The set of all possible values of an attribute, such as integers from 0 to 100 for a grade, is the attribute **domain**. In an **ER model**, an attribute name appears in an oval that has a line to the corresponding entity box, such as in Figure 3. ...  
The set of all possible values of an attribute is the attribute **domain**.



# Relational Algebra operations

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

- Select Operation ( $\sigma$ )
- It selects tuples that satisfy the given predicate from a relation.
- **Notation** –  $\sigma_p(r)$
- Where  $\sigma$  stands for selection predicate and  $r$  stands for relation.  $p$  is propositional logic formula which may use connectors like **and**, **or**, and **not**. These terms may use relational operators like  $=$ ,  $\neq$ ,  $\geq$ ,  $<$ ,  $>$ ,  $\leq$ .
- **For example** –
- $\sigma_{subject = "database"}(\text{Books})$  **Output** – Selects tuples from books where subject is 'database'.

- Project Operation ( $\Pi$ )
- It projects column(s) that satisfy a given predicate.
- Notation –  $\Pi_{A_1, A_2, A_n} (r)$
- Where  $A_1, A_2, A_n$  are attribute names of relation  $r$ .
- Duplicate rows are automatically eliminated, as relation is a set.
- **For example –**
- $\Pi_{\text{subject, author}} (\text{Books})$  Selects and projects columns named as subject and author from the relation Books.

- Union Operation (U)
- It performs binary union between two given relations and is defined as –
- $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$  **Notation** –  $r \cup s$
- Where **r** and **s** are either database relations or relation result set (temporary relation).
- For a union operation to be valid, the following conditions must hold –
- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.
- $\Pi_{\text{author}}(\text{Books}) \cup \Pi_{\text{author}}(\text{Articles})$  **Output** – Projects the names of the authors who have either written a book or an article or both.

- Set Difference ( $-$ )
- The result of set difference operation is tuples, which are present in one relation but are not in the second relation.
- **Notation** –  $r - s$
- Finds all the tuples that are present in  $r$  but not in  $s$ .
- $\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$  **Output** – Provides the name of authors who have written books but not articles.



- Cartesian Product ( $\times$ )
- Combines information of two different relations into one.
- **Notation** –  $r \times s$
- Where  $r$  and  $s$  are relations and their output will be defined as –
- $r \times s = \{ q \ t \mid q \in r \text{ and } t \in s \}$
- $\sigma_{\text{author} = \text{'tutorialspoint'}}(\text{Books} \times \text{Articles})$  **Output** – Yields a relation, which shows all the books and articles written by tutorialspoint.

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5273**

**@gmail.com**



**S.P. Mandali Pune  
Prin. K. P. Mangalvedhekar Institute of  
Management Career Development and  
Research.**

**156-B Railway Lines Solapur.  
Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject DBMS  
Programme:- BCAII Sem IV**

## Introduction to database management System

- Concept of Data and Information
- Data can be defined as a collection of facts and figures.
- Each organisation has to record the details of such transaction and use it for further processing like preparing balance sheet, profit and loss a/c
- Such a processing is done on data it is called information So information is a processed data
- Which helps the managers to make decision
  
- Data
- Data is collection of facts No Conclusion can be drawn on data . Data can not be used for decision making. Data is a raw material e.g 1
  
- Information
- Information is a processed form of data .Information can be used to draw conclusion. Information plays an important role in decision making .information is just like a finished product e.g. roll no 1

## Need of Database Management

- The term database can be defined as a collection of interrelated data of an organisation . Any organization has to maintain the records of all its activities which are interdependent e.g. The purchase department has to maintain records of purchases and plan for future purchase which depend on the current stock position maintained by stores department Production department produces finished product according to the sales orders provided by Sales department Production department uses raw material in stores department for production
  
- So unless Stock department is given a report of items with reorder level stock Purchase department can not purchase material. .As the stock has reached at minimum reorder level

.Production department can not proceed with production and sales department can not fulfil the sales order



- Thus if any organization fails to produce proper reports . The whole organisation can not run smoothly
- The above example shows the need of database management system
- The database of an organisatin must be maintained in such a way that it can be used to produce effective reports at times
- Constructing the database: Insertion,Updation and Deletion of data
- Manipulating the database:- Querying the database and Generating the reports Role

of Database Management System

- Constructing the database: Insertion,Updation and Deletion of data
- Manipulating the database:- Querying the database and Generating the reports

Limitations of traditional file system

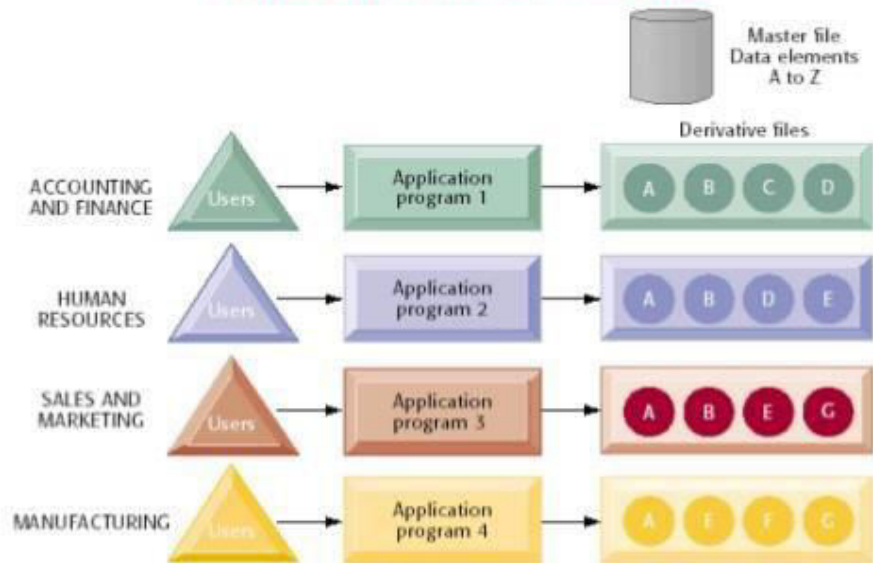
- 1) Data Redundancy and Inconsistency
- 2) Difficulty in Accessing data
- 3) Integrity problems

- 4) Concurrent Access Problems
- Security Problems

**Management Information Systems**

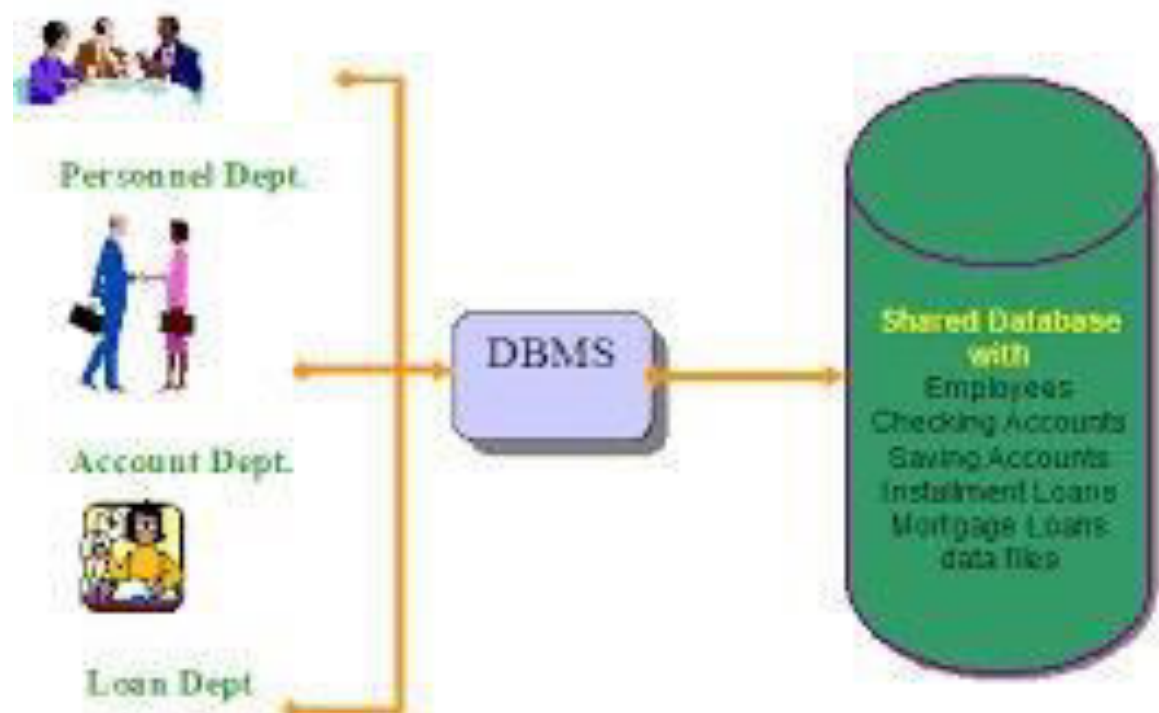
**ORGANIZING DATA IN A TRADITIONAL FILE ENVIRONMENT**

**Traditional File Processing**



7.10

© 2006 by Prentice Hall







**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5  
273**

**@gmail.com**

# ASP.NET

By Deshpande M.K.

# What is ASP.Net?

- ASP.Net is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web application for PC, as well as mobile devices.
- ASP.Net works on top of the HTTP protocol and uses the HTTP commands and policies to set a browser-to-server two-way communication and cooperation.
- ASP.Net is a part of Microsoft .Net platform. ASP.Net applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.
- The ASP.Net application codes could be written in either of the following languages:
  - C#
  - Visual Basic .Net
  - Jscript
  - J#
- ASP.Net is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls like text boxes, buttons and labels for assembling, configuring and manipulating code to create HTML pages.

# ASP.Net Web Forms Model

- ASP.Net web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.
- All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.
- Now, HTTP is a stateless protocol. ASP.Net framework helps in storing the information regarding the state of the application, which consists of:
  - Page state
  - Session state
- The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective obtained from various pages the user visited and worked with, i.e., the overall session state. To clear the concept, let us take up an example of a shopping cart as follows.
- User adds items to a shopping cart. Items are selected from a page, say the items page, and the total collected items and price are shown in a different page, say the cart page. Only HTTP cannot keep track of all the information coming from various pages. ASP.Net session state and server side infrastructure keeps track of the information collected globally over a session.
- The ASP.Net runtime carries the page state to and from the server across page requests while generating the ASP.Net runtime codes and incorporates the state of the server side components in hidden fields.
- This way the server becomes aware of the overall application state and operates in a two-tiered connected way.

# ASP.Net Component Model:

- The ASP.Net component model provides various building blocks of ASP.Net pages. Basically it is an object model, which describes:
  - Server side counterparts of almost all HTML elements or tags, like `<form>` and `<input>`.
  - Server controls, which help in developing complex user-interface for example the Calendar control or the Gridview control.
- ASP.Net is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.Net web application is made of pages. When a user requests an ASP.Net page, the IIS delegates the processing of the page to the ASP.Net runtime system.
- The ASP.Net runtime transforms the .aspx page into an instance of a class, which inherits from the base class Page of the .Net framework. Therefore, each ASP.Net page is an object and all its components i.e., the server-side controls are also objects.

# Components of .Net Framework

---

- Before going to the next session on Visual Studio.Net, let us look at the various components of the .Net framework 3.5. The following table describes the components of the .Net framework 3.5 and the job they perform:



# Visual Studio IDE

- ASP.Net provides an abstraction layer on top of HTTP on which the web applications are built. It provides high-level entities like classes and components within an object-oriented paradigm.
- The key development tool for building ASP.Net applications and front ends is Visual Studio. In these tutorials, we will work on Visual Studio 2008.
- Visual Studio is an integrated development environment for writing, compiling and debugging the code. It provides a complete set of development tools for building ASP.Net web applications, web services, desktop applications and mobile applications.

# Visual Studio IDE

- When you start a new web site, ASP.NET provides the starting folders and files for the site, including two files for the first web form of the site.
- The file named Default.aspx contains the HTML and asp code that defines the form, and the file named Default.aspx.cs (for C# coding) or the file named Default.aspx.vb (for vb coding) contains the code in the language you have chosen and this code is responsible for the form's works.
- The primary window in the Visual Studio IDE is the Web Forms Designer window. Other supporting windows are the Toolbox, the Solution Explorer, and the Properties window. You use the designer to design a web form, to add code to the control on the form so that the form works according to your need, you use the code editor.



# Ways to work with views and windows

The following are the ways to work with different windows:

- To change the Web Forms Designer from one view to another, click on the Design or source button.
- To close a window, click on the close button on the upper right corner and to redisplay, select it from the View menu.
- To hide a window, click on its Auto Hide button; the window changes into a tab, to redisplay again click on the Auto Hide button again.
- To size a window just drag it.

# Projects and Solutions

- A typical ASP.Net application consists of many items: the web content files (.aspx), source files (e.g., the .cs files), assemblies (e.g., the .dll files and .exe files), data source files (e.g., .mdb files), references, icons, user controls and miscellaneous other files and folders. All these files that make up the website are contained in a Solution.
- When a new website is created VB2008 automatically creates the solution and displays it in the solution explorer.
- Solutions may contain one or more projects. A project contains content files, source files, and other files like data sources and image files. Generally the contents of a project are compiled into an assembly as an executable file (.exe) or a dynamic link library (.dll) file.
- Typically a project contains the following content files:
  - Page file (.aspx)
  - User control (.ascx)
  - Web service (.asmx)
  - Master page (.master)
  - Site map (.sitemap)
  - Website configuration file (.config)

# Building and Running a Project:

---

- The application is run by selecting either Start or Start Without Debugging from the Debug menu, or by pressing F5 or Ctrl-F5. The program is built i.e. the .exe or the .dll files are generated by selecting a command from the Build menu.

# ASP.Net Page Life Cycle

Following are the different stages of an ASP.Net page:

- **Page request** . when ASP.Net gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent
- **Starting of page life cycle** . at this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
- **Page initialization** . at this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.
- **Page load** . at this stage, control properties are set using the view state and control state values.
- **Validation** . Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.
- **PostBack event handling** . if the request is a postback (old request), the related event handler is called.
- **Page rendering** . at this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.
- **Unload** . the rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

# **Hibernate Framework**

**Created by Rapelli**

# Objective

To understand hibernate framework and how to code using hibernate framework

# Limitations of JDBC

- Boiler plate code
- Paradigm mismatch
- Developer needs to know database table  
Inheritance can not be implemented
- Association can not be implemented
-

# What Is The Solution

- Hibernate
  - Latest Open Source Persistence technology
  - Maps Java classes with database tables
  - Reduced development time for data queries & retrieval of data



# How Data Can Be Stored Using Java?

```
public class Employee
{
private int id;
private String
    first_name;
private String
    last_name;
private int salary;
public Employee() {}
```

```
public Employee(String fname,
String lname, int salary)
{
this.first_name    =    fname;
this.last_name     =    lname;
this.salary = salary;
}
public int getId()
    { return id; }
public String getFirstName()
    { return first_name; }
public String getLastName() { return
    last_name; }
public int getSalary() { return salary;}}
```

# How data can be stored using RDBMS?

- create table EMPLOYEE ( id INT NOT NULL auto\_increment, first\_name VARCHAR(20) default NULL, last\_name VARCHAR(20) default NULL, salary INT default NULL, PRIMARY KEY (id) );

# MVC design pattern

- This design pattern helps us develop loosely coupled application by segregating various concerns into different layers.
- MVC design pattern enforces the application to be divided into three layers, Model, View and Controller.

# What Problems May Occur?

Mismatch	Description
Granularity	Sometimes you will have an object model which has more classes than the number of corresponding tables in the database
Inheritance	RDBMSs do not define anything similar to Inheritance which is a natural paradigm in object-oriented programming languages
Identity	A RDBMS defines exactly one notion of 'sameness': the primary key. Java, however, defines both object identity ( <code>a==b</code> ) and object equality ( <code>a.equals(b)</code> )
Associations	Object-oriented languages represent associations using object references where as an RDBMS represents an association as a foreign key column
Navigation	The ways you access objects in Java and in a RDBMS are fundamentally different

# Solution

- ORM
- The Object-Relational Mapping (ORM) is the solution to handle all the mismatches
- A programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc

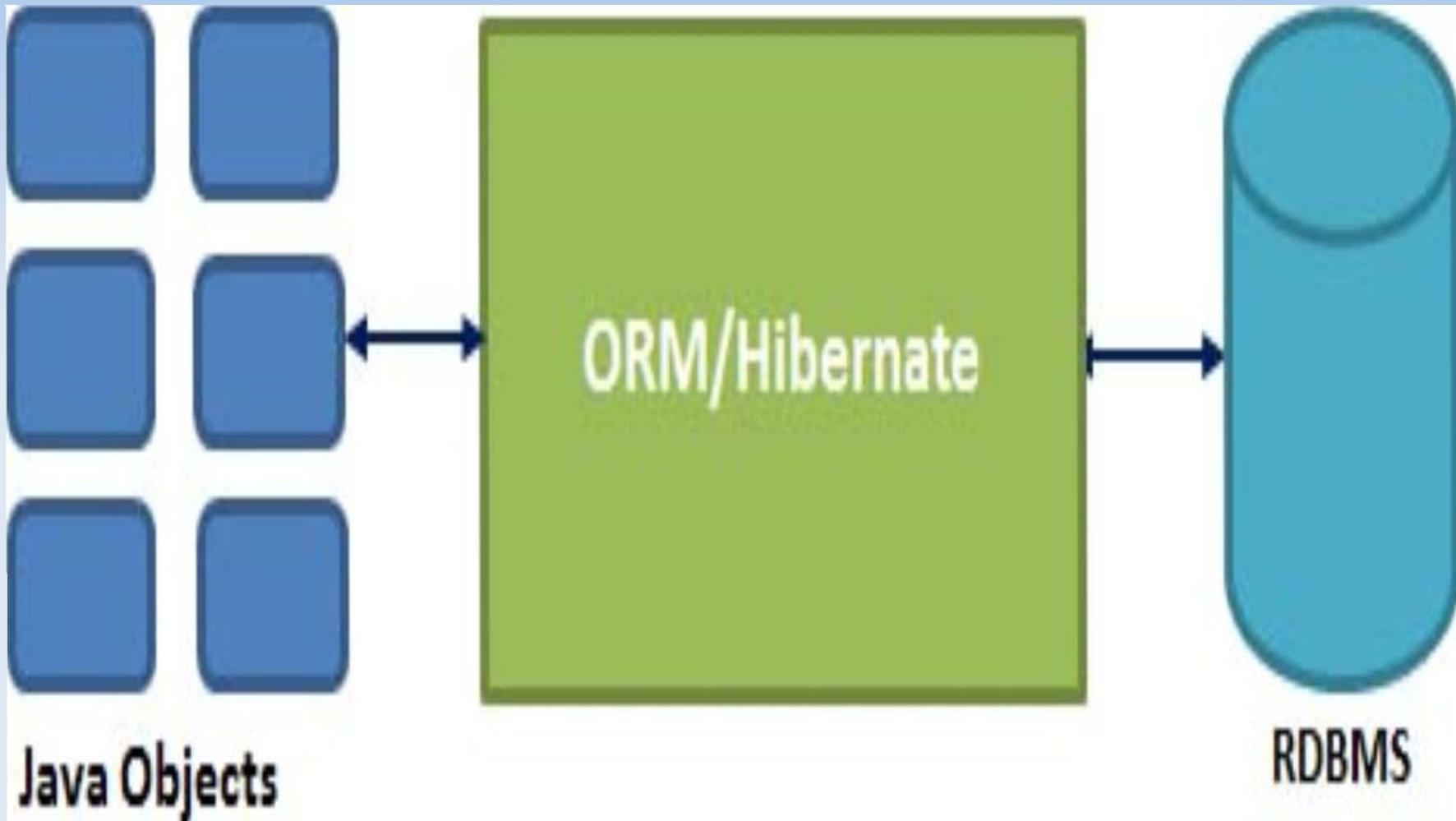
# Java ORM Frameworks

- A persistent framework is an ORM service that stores and retrieves objects into a relational database
  - Enterprise JavaBeans Entity Beans
  - Java Data Objects
  - Castor
  - TopLink
  - Spring DAO
  - Hibernate

# Hibernate

- Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework
- It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application
- Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks

# What Hibernate does?

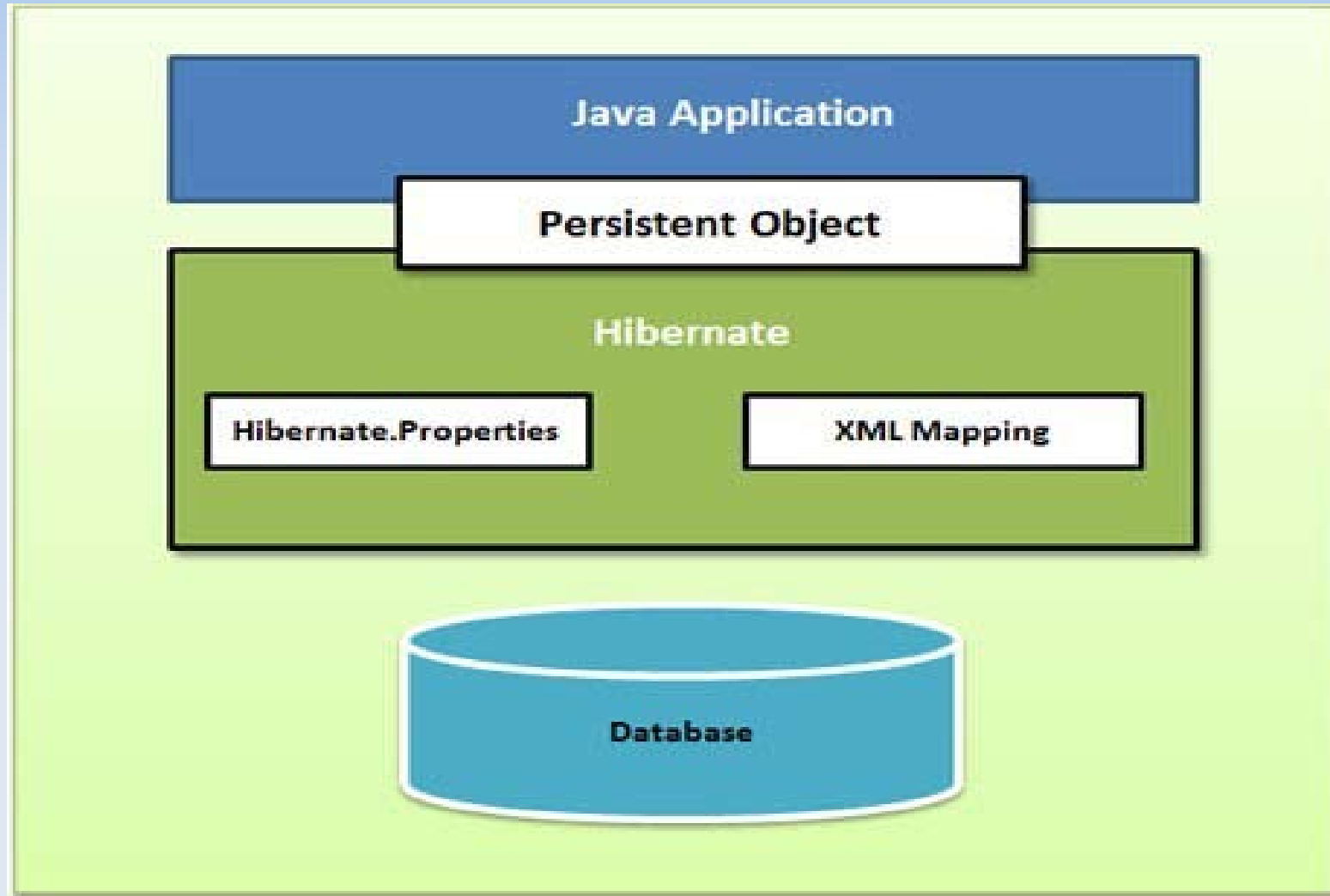




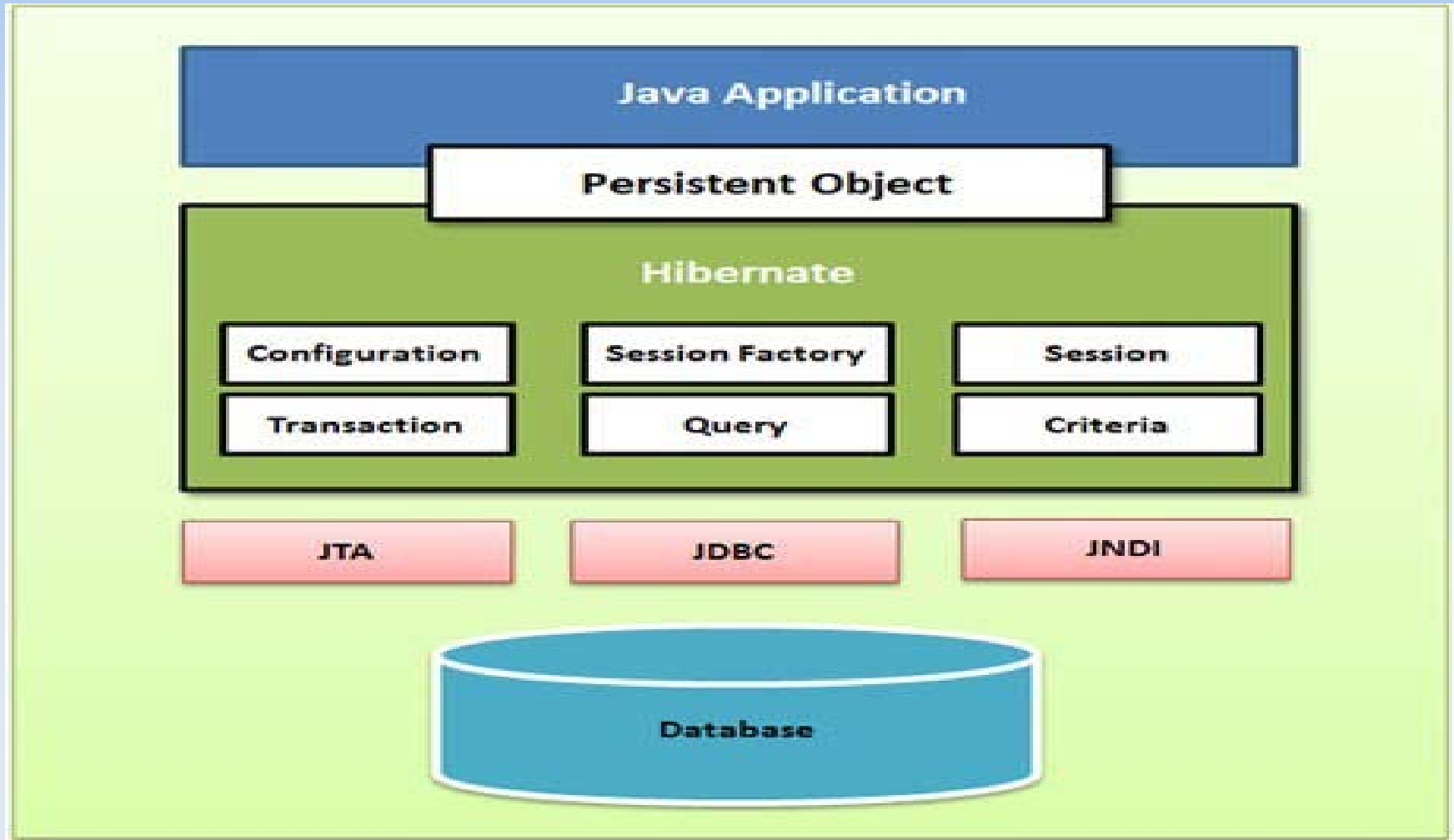
# Hibernate Advantages

- Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code
- Provides simple APIs for storing and retrieving Java objects directly to and from the database
- If there is change in Database or in any table then the only need to change XML file properties
- Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects
- Hibernate does not require an application server to operate
- Manipulates Complex associations of objects of your database
- Minimize database access with smart fetching strategies
- Provides Simple querying of data

# Hibernate Architecture (high level)



# Hibernate Architecture (details)



# Configuration Object

- The Configuration object is the first Hibernate object created in any Hibernate application
- Created only once during application initialization
- It represents a configuration or properties file required by the Hibernate

# SessionFactory Object

- Configuration object is used to create a SessionFactory object
- It configures Hibernate for the application using the supplied configuration file
- A Session object is instantiated using SessionFactory

# Session Object

- A Session is used to get a physical connection with a database
- This is lightweight object
- Needs to be instantiated each time an interaction is needed with the database
- Persistent objects are saved and retrieved through a Session object
- These are not thread safe

# Transaction Object

- A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality
- Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA)

# Query Object

- Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects
- A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query



# Criteria Object

- Criteria object are used to create and execute object oriented criteria queries to retrieve objects
- Its an alternative to HQL & SQL

# What properties in .cfg.xml file ?

- **hibernate.dialect**

This property makes Hibernate generate the appropriate SQL for the chosen database

## **hibernate.connection.driver\_class**

- The JDBC driver class

## **hibernate.connection.url**

- The JDBC URL to the database instance

## **hibernate.connection.username**

- The database username

# Hibernate.cfg.xml file

```
<hibernate-configuration>
```

```
<session-factory>
```

```
<property
```

```
name="hbm2ddl.auto">update</property>
```

```
<property
```

```
name="dialect">org.hibernate.dialect.MySQL
```

```
Dialect</property> <property
```

```
name="connection.url">jdbc:mysql://localho
```

```
st:3306/ADJT</property>
```

```
<property
```

```
name="connection.username">root</proper
```

# Example pojo.hbm.xml file

```
<hibernate-mapping>  
  <class  
    name="com.javatpoint.mypackage.Employee"  
    table="emp">  
    <id name="id">  
      <generator class="assigned"></generator>  
    </id>  
    <property name="firstName"></property>  
    <property name="lastName"></property>  
  </class>
```

# Generator

- Types of Generators
  - ◆ Assigned
  - ◆ Increment
  - ◆ Sequence
  - ◆ Identity
  - ◆ Foreign
  - ◆ hillo native
  - ◆

# Assigned

Supported by all DBs

- Default generator type
- Developer has to specify which PK value is assigned to object i.e. which column will be used as PK

```
<id name="proId" column="pid">
```

```
<generator/>
```

```
</id>
```

# Increment

- Supported by all DBs
- Generates the id for new record by using formula as max of id value in database plus 1

# Sequence

- Not supported by MySQL
- First ensures if DB supports this.
- If yes then while inserting a record, hibernate gets next value from the sequence.
- If programmer has created sequence that can be used as

```
<id name="productId" column="pid">
```

```
  <generator>
```

```
    <param
```



# hilo

- This generator is database independent
- for the first record, the id value will be inserted as 1
- actually this hibernate stores the count of id values generated in a column of separated table, with name “hibernate\_unique\_key” by default with the column name “next\_hi”

```
<id name="productId" column="pid">
```

```
<generator class="hilo">
```

```
<param name="table">your table
```

```
name</param>
```

# Identity

- Not supported by Oracle
- Id value is generated by hibernate database & not

# Foreign

- Used in one-to-one mapping

# What is Component Mapping ?

- A **Component** mapping is a mapping for a class having a reference to another class as a member variable

**e.g. Employee has address**

```
create table EMPLOYEE ( id INT NOT NULL
auto_increment, first_name VARCHAR(20)
default NULL, last_name VARCHAR(20)
default NULL, salary INT default NULL,
street_name VARCHAR(40) default NULL,
city_name VARCHAR(40) default NULL,
state_name VARCHAR(40) default NULL,
zipcode VARCHAR(10) default NULL
```

# What is Component Mapping ?

- we will map the dependent object as a component.
- Mapping:

```
<component name="address"  
class="com.Address">
```

```
<property name="city"></property>
```

```
<property name="country"></property>
```

```
<property name="pincode"></property>
```

```
</component>
```

# Inheritance Mapping

There are three types of Inheritance Mapping

- Table per concrete class
- Table per sub class
- Table per class hierarchy (Single table Strategy)

There are three types of inheritance mapping in hibernate

1. Table per concrete class with unions

2. Table per class hierarchy (Single table Strategy)

3. Table per subclass

# Table per class Hierarchy

- Single Table can be mapped to a class hierarchy
- Mapping as

```
<discriminator column="type"  
  type="string"></discriminator>
```

```
<property name="name"></property>
```

```
<subclass name="mypackage.Regular_Employee"  
  discriminator-value="reg_emp">
```

```
<property name="salary"></property>
```

```
<property name="bonus"></property>
```

```
</subclass>
```

# Table per class Hierarchy

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
TYPE	VARCHAR2(255)	No	-	-
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 7



# Table per class Hierarchy

## Advantages of Single Table per class hierarchy

- Simplest to implement.
- Only one table to deal with.
- Performance wise better than all strategies because no joins or sub-selects need to be performed.

## Disadvantages:

- Most of the column of table are nullable so the NOT NULL constraint cannot be applied.

# Table per concrete class

- It creates separate table per concrete class
- Limitation – Redudancy code
- Mapping is

```
<union-subclass
```

```
  name="mypackage.Contract_Employee"
```

```
  table="contemp122">
```

```
<property name="pay_per_hour"></property>
```

```
  <property
```

```
    name="contract_duration"></property> </union-
```

```
subclass>
```

# Table per concrete class

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
				1 - 2

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
				1 - 4

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 4

# Table per subclass

- Here, we have separate **Parent** table holding common properties & subclass table points to it using **Foreign Key**.

- Mapping as

```
<joined-subclass
```

```
  name="mypackage.Contract_Employee"
```

```
  table="contemp123">
```

```
  <key column="eid"></key>
```

```
  <property name="pay_per_hour"></property>
```

```
  <property
```

# Table per subclass

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
				1 - 2

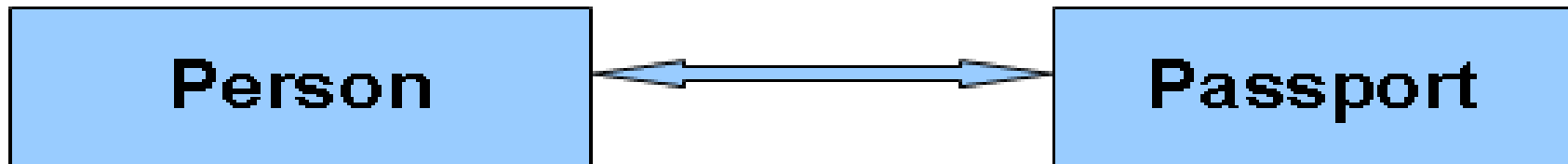
Column Name	Data Type	Nullable	Default	Primary Key
EID	NUMBER(10,0)	No	-	1
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
				1 - 3

Column Name	Data Type	Nullable	Default	Primary Key
EID	NUMBER(10,0)	No	-	1
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 3

# Hibernate Association (Collection Mapping)

- In hibernate, Hibernate Association is the concept where we putting relationship between two entities.
- Using hibernate we can put the following relationship;
  - One-to-One
  - One-to-Many
  - Many-to-Many

# One to One mapping



**Person Table**

ID	INT(10)
passport_id	INT(10)
First_name	VARCHAR(50)
last_name	VARCHAR(50)
dob	VARCHAR(50)

**Passport Table**

ID	INT(10)
passport_number	VARCHAR(20)
valid_date	DATE



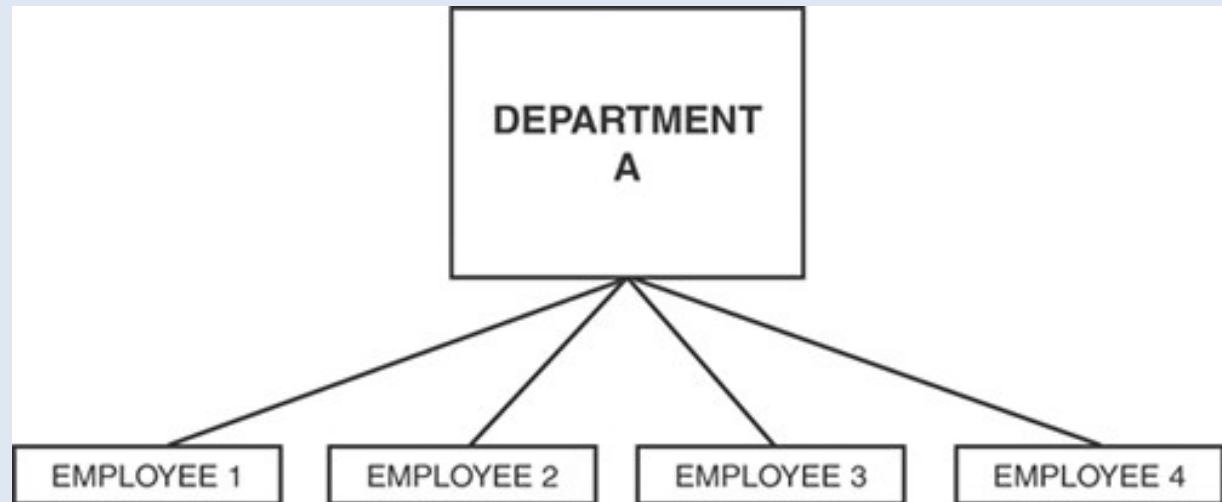
# One to One mapping

- Attributes used in pojo.hbm.xml file
- **name**: the name of the property
- **class** (defaults to the property type determined by reflection): the name of the associated class
- **cascade** (optional): specifies which operations should be cascaded from the parent object to the associated object



# One to Many Mapping

- A relationship in which each record in one table is linked to multiple records in another table
- E.g. One department can have many employees



# One to Many Example

- Employee.hbm.xml
- **Dept.hbm.xml**

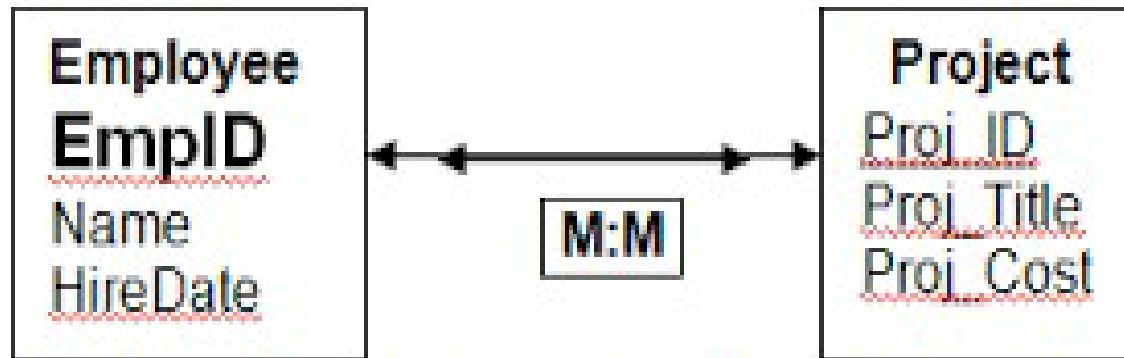
.....

```
<set name="employees" cascade="all" >  
<key column="deptId" />  
<one-to-many class="com.Employee" />  
</set>
```

.....

# Many to Many Mapping

■



**Fig. 1.10: Example of a Many-to-Many relationship**

# Many to Many Mapping

- A **Many-to-Many** mapping can be implemented using a **Set** java collection that does not contain any duplicate element
- E.g. many employees involved in many meetings
- Many meetings having many employees

# HQL

- Hibernate Query Language (HQL) is an **object-oriented query language**, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties.
- HQL queries are translated by Hibernate into conventional SQL queries which in turns perform action on database.

# Query Interface

- It is an object oriented representation of Hibernate Query.

```
Query hql=session.createQuery("from table");
```

- **Query Interface methods:**

**1.public int executeUpdate()** is used to execute the update or delete query.

**2.public Query setFirstResult(int rowno)** specifies the row number from where record will be retrieved.

# Query Interface

**3.public List list()** returns the result of the relation(table) as a list.

**4.public Query setMaxResult(int rowno)** specifies the no. of records to be retrieved from the relation (table).

**5.public Query setParameter(int position, Object value)** it sets the value to the JDBC style query parameter.

**6.public Query setParameter(String name, Object value)** it sets the value to a named query parameter.

# Syntax

```
Query query=session.createQuery("from Emp");  
    //here persistent class name is Emp List  
list=query.list();
```



# Clauses in HQL are:

- **From Clause:** It's same as select clause in SQL, from Employee is same as select \* from Employee.  
We can also create alias such as from **Employee** **emp** or from Employee as emp.
- **Join Clause:** HQL supports inner join, left outer join, right outer join and full join. For example, select e.name, a.city from Employee e INNER JOIN e.address.
- **Aggregate Functions:** HQL supports commonly used aggregate functions such as count(\*), count(distinct x), min(), max(), avg() and sum().

## Clauses in HQL are:

- **AS Clause**

AS clause can be used to assign aliases to the classes in your HQL queries

```
String hql = "FROM Employee AS E";
```

- **SELECT Clause**

The SELECT clause provides more control over the result set than the from clause.

```
String hql = "SELECT E.firstName FROM  
Employee E";
```

## HQL :

- **Expressions:** HQL supports arithmetic expressions (+, -, \*, /), binary comparison operators (=, >=, <=, <>, !=, like), logical operations (and, or, not) etc.
- HQL also supports order by and group by clauses.
- HQL also supports sub-queries just like SQL queries.
- HQL supports DDL, DML and executing store procedures too.

# HQL update query

```
Transaction tx=session.beginTransaction(); Query  
q=session.createQuery("update User set  
name=:n where id=:i");  
q.setParameter("n","Kumar");  
q.setParameter("i",100);  
  
int status=q.executeUpdate();  
System.out.println(status);  
tx.commit();
```

# HQL delete query

```
Query query=session.createQuery("delete fr om  
    Emp where id=100");  
//specifying class name (Emp)  
not tablename      query.executeUpdate();
```

# Hibernate - Criteria Queries

- Hibernate provides alternate ways of manipulating objects and in turn data available in RDBMS tables.
- The Hibernate **Session** interface provides **createCriteria()** method which can be used to create a **Criteria** object that returns instances of the persistence object's class when your application executes a criteria query.

Example :

- `Criteria cr = session.createCriteria(Employee.class);` List
- `results = cr.list();`

# Restrictions with Criteria

- You can use **add()** method available for **Criteria** object to add restriction for a criteria query.
- Example to add a restriction to return the records with salary is equal to 2000:
- Criteria cr =  
session.createCriteria(Employee.class);  
cr.add(Restrictions.eq("salary", 2000));  
List results = cr.list();

# Criteria examples

```
Criteria cr = session.createCriteria(Employee.class);  
// To get records having salary more than 2000  
cr.add(Restrictions.gt("salary", 2000));
```

```
// To get records having salary less than 2000  
cr.add(Restrictions.lt("salary", 2000));
```

```
// To get records having firstName starting with zara  
cr.add(Restrictions.like("firstName", "zara%"));
```

```
// To get records having salary in between 1000 and 2000  
cr.add(Restrictions.between("salary", 1000, 2000));
```

```
// To check if the given property is empty  
cr.add(Restrictions.isEmpty("salary"));
```



# HQL vs Criteria

- HQL is to perform both select and non-select operations on the data, but Criteria is only for selecting the data, we cannot perform non-select operations using criteria
- HQL is suitable for executing Static Queries, where as Criteria is suitable for executing Dynamic Queries
- Criteria used to take more time to execute then HQL
- With Criteria we are safe with SQL Injection because of its dynamic query generation but in HQL as your queries are either fixed or parametrized, there is no safe from SQL Injection.

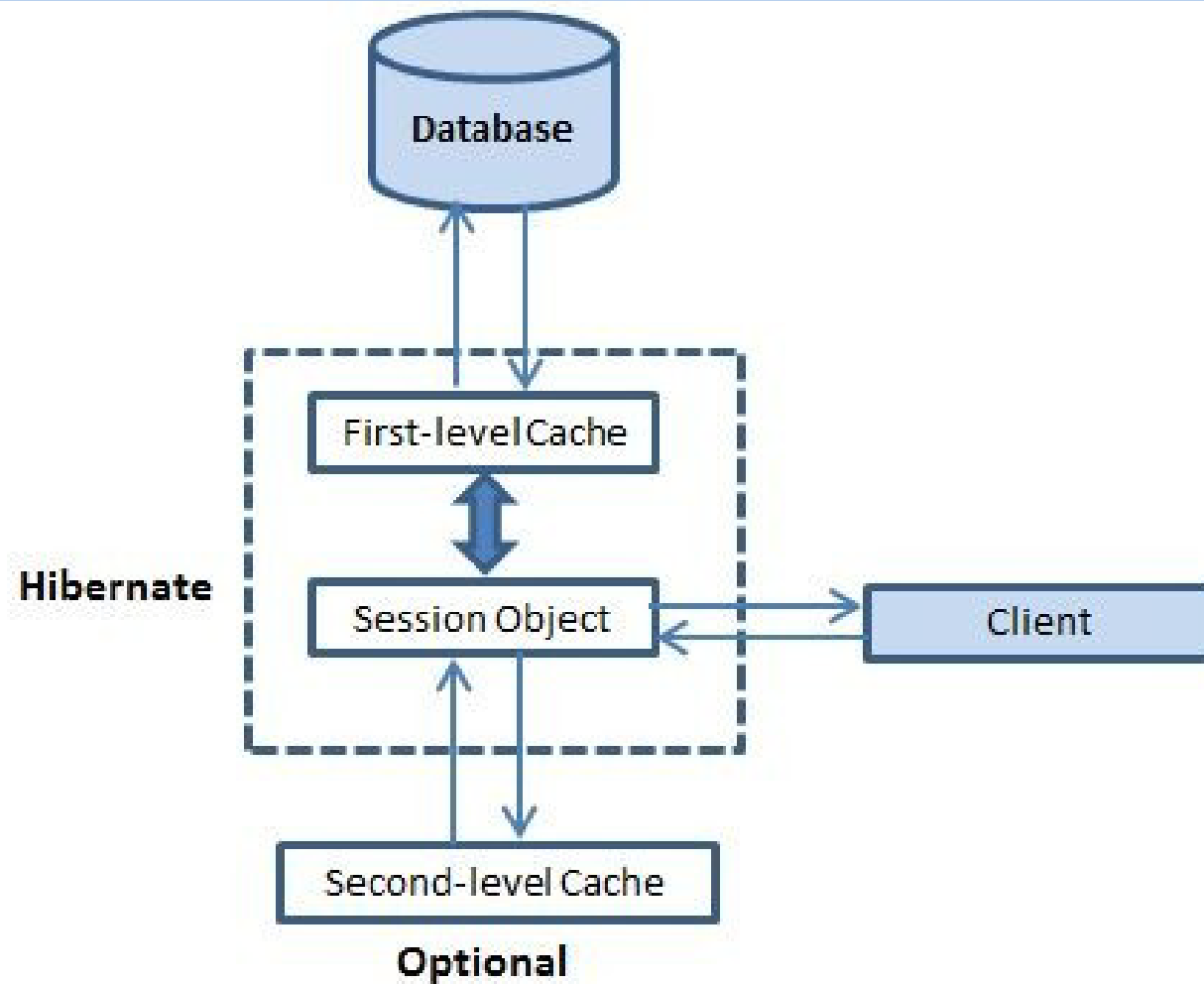
# HQL vs Criteria

HQL	Criteria
It performs both select and non select operation	It performs only select operation
It is suitable for executing static queries	It is suitable for executing dynamic queries
it is faster	it is slower than HQL
Non safe SQL injection	safe SQL injection

# Caching In Hibernate

- Caching is a mechanism to enhance the performance of a system.
- It is a buffer memory that lies between the application and the database.
- Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

# Caching...



# caching

Hibernate provide two level cache

- The first-level cache - Session
- The second-level cache -SessionFactory-level cache

The first-level cache - Session (Earlier hibernate already provide this level of cache)

The second-level cache -Session-factory-level cache

# First Level cache

- The first level cache type is the **session** cache.
- The session cache caches object within the current session but this is not enough for long level i.e. session factory scope.
- It is by default enable & can't disable manually.

# Second level cache

- It is optional
- The second level cache can be configured on a per-class and per-collection basis and mainly responsible for caching objects across sessions.
- Any third-party cache can be used with Hibernate.
- Different vendors have provided the implementation of Second Level Cache.
  - EH Cache
  - OS Cache
  - Swarm Cache

# Second level cache

- It is by default disabled & can be enabled by;
  - `<!-- Enable the second-level cache -->`
  - `<property  
name="cache.use_second_level_cache">true</property>`
  - `<property  
name="cache.provider_class">org.hibernate.  
cache.EhCacheProvider</property>`



# Second level cache

- **Cache usage functionality:**

**read-only:** caching will work for read only operation.

**nonstrict-read-write:** caching will work for read and write but one at a time.

**read-write:** caching will work for read and write, can be used simultaneously.

**transactional:** caching will work for transaction.

**Ex:** `<cache usage="read-only" />`

# Second level cache

- Every cache provider is not compatible with every concurrency strategy.

Strategy/Provider	Read-only	Nonstrictread-write	Read-write	Transactional
<b>EHCache</b>	X	X	X	
<b>OSCache</b>	X	X	X	
SwarmCache	X	X		
JBoss Cache	X			X



**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of**  
**Management Career Development and**  
**Research.**

**156-B Railway Lines Solapur.**  
**Affiliated PAH Solapur University**

Name of Faculty . Mr Santosh Kulkarni

Subject Data Mining  
Programme:- BCAIII Sem V

Name of Faculty :-Mr.Santosh kulkarni

Subject : Data Mining

Course :- BCA III Sem V

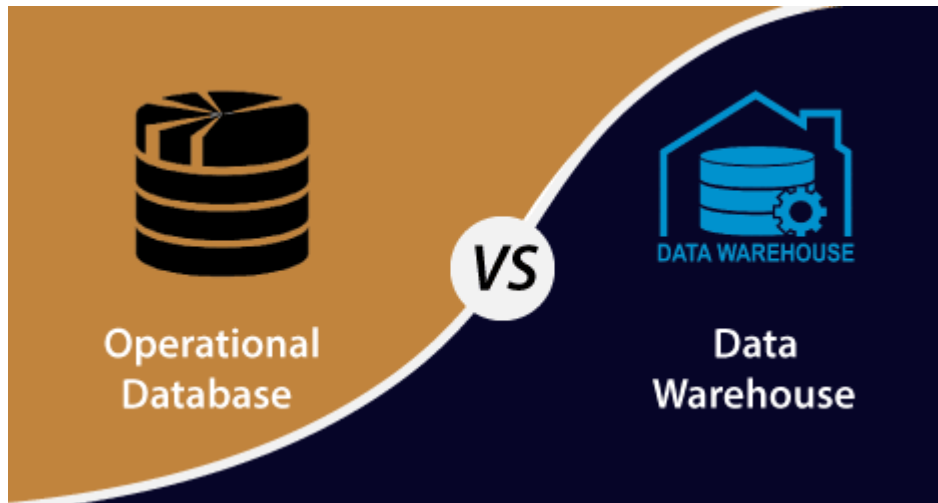
Data and Information

# Introduction to **DATA WAREHOUSING**



A data warehouse is constructed by integrating data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries and decision making. This tutorial adopts a step-by-step approach to explain all the necessary concepts of data warehousing

Difference between operational database vs data warehouse



- The Operational Database is the source of information for the data warehouse. It includes detailed information used to run the day to day operations of the business. The data frequently changes as updates are made and reflect the current value of the last transactions.
- Operational Database Management Systems also called as OLTP (Online Transactions Processing Databases), are used to manage dynamic data in real-time.
- Data Warehouse Systems serve users or knowledge workers in the purpose of data analysis and decision-making. Such systems can organize and present information in specific formats to accommodate the diverse needs of various users. These systems are called as Online-Analytical Processing (OLAP) Systems.

Characterstics of Data Warehouse

- **Subject-oriented –**

A data warehouse is always a subject oriented as it delivers information about a theme instead of organization's current operations. It can be achieved on specific theme. That means the data warehousing process is proposed to handle with a specific theme which is more defined. These themes can be sales, distributions, marketing etc.

A data warehouse never put emphasis only current operations. Instead, it focuses on demonstrating and analysis of data to make various decision. It also delivers an easy and precise demonstration around particular theme by eliminating data which is not required to make the decisions.

- **Integrated –**

It is somewhere same as subject orientation which is made in a reliable format. Integration means founding a shared entity to scale the all similar data from the different databases. The data also required to be resided into various data warehouse in shared and generally granted manner.

A data warehouse is built by integrating data from various sources of data such that a mainframe and a relational database. In addition, it must have reliable naming conventions, format and codes. Integration of data warehouse benefits in effective analysis of data. Reliability in naming conventions, column scaling, encoding structure etc. should be confirmed. Integration of data warehouse handles various subject related warehouse.

- **Time-Variant –**

In this data is maintained via different intervals of time such as weekly, monthly, or annually etc. It finds various time limit which are structured between the large datasets and are held in online transaction process (OLTP). The time limits for data warehouse is wide-ranged than that of operational systems. The data resided in data warehouse is predictable with a specific interval of time and delivers information from the historical perspective. It comprises elements of time explicitly or implicitly. Another feature of time-variance is that once data is stored in the data warehouse then it cannot be modified, alter, or updated.

- **Non-Volatile –**

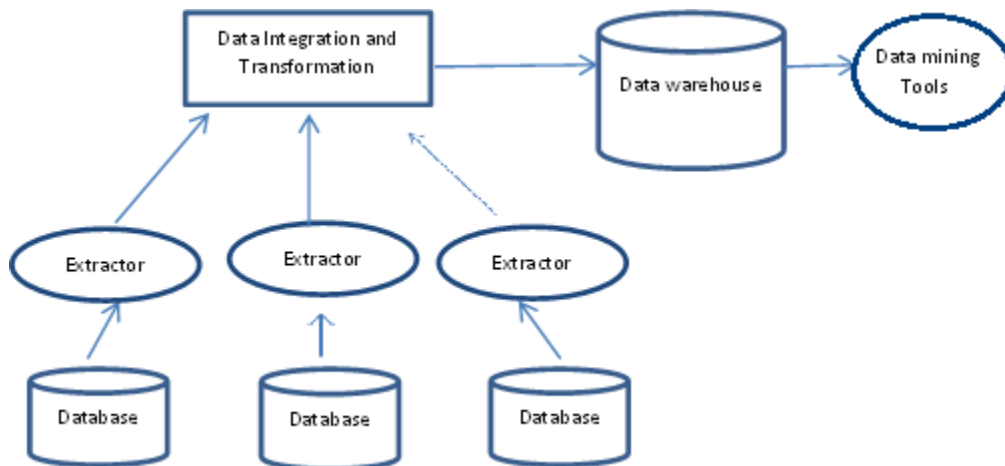
As the name defines the data resided in data warehouse is permanent. It also means that data is not erased or deleted when new data is inserted. It includes the mammoth quantity of data that is inserted into modification between the selected quantity on logical business. It evaluates the analysis within the technologies of warehouse.

#### Architectural Types

1. 1) Centralised Data Warehouse
2. Independent Data Marts
3. Federated
4. Hub and Spoke
5. Data Mart Bus



## 1) Centralised Data Warehouse



**Centralized Data Warehouse.** A **Centralized Data Warehouse** is a **data warehousing** implementation wherein a single **data warehouse** serves the needs of several separate business units simultaneously using a single **data** model that spans the needs of multiple business divisions

## 2) Independent Data Marts

This Architect type evolves in organisation where the organisational units develop their own data marts for their own specific purposes

## 3) Federated Data Marts

Some Organisations get into data warehousing with an existing decision support structures in the form of operational systems ,extracted data sets,primitive data marts

#### 4) Hub and Spoke

This is the similar centralised data warehouse architecture

Dependent data marts obtain data from centralised data warehouse

The centralised data warehouse forms the hub to feed data to the data marts on the spokes

#### 5) Data Mart Bus

In this architecture begin with analysing requirements for a specific business objects

Such as orders, billings, shipments

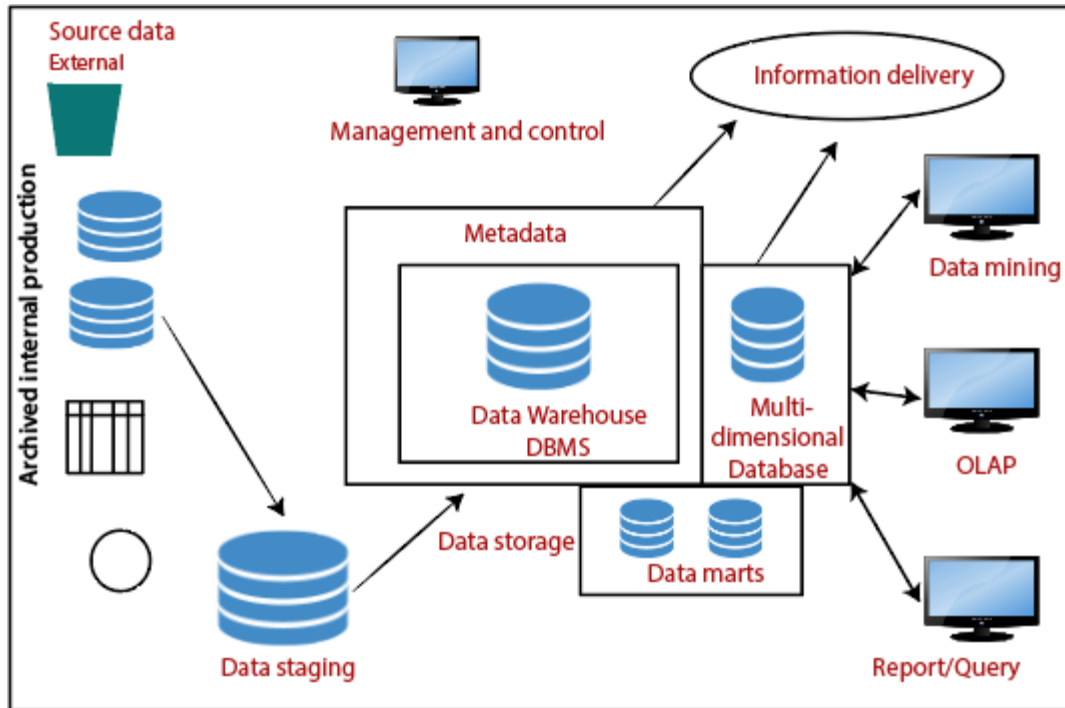
Build the data mart using business dimensions

## Components or Building Blocks of Data Warehouse

Architecture is the proper arrangement of the elements. We build a data warehouse with software and hardware components. To suit the requirements of our organizations, we arrange these building we may want to boost up another part with extra tools and services. All of these depends on our circumstances.

The figure shows the essential elements of a typical warehouse. We see the Source Data component shows on the left. The Data staging element serves as the next building block. In the middle, we see the Data Storage component that handles the data warehouses data. This element not only stores and manages the data; it also keeps track of data using the metadata repository. The Information Delivery component shows on the right consists of all the different ways of making the information from the data warehouses available to the users.

### Source Data Component



**Components or Building Blocks of Data Warehouse**

Source data coming into the data warehouses may be grouped into four broad categories:

**Production Data:** This type of data comes from the different operating systems of the enterprise. Based on the data requirements in the data warehouse, we choose segments of the data from the various operational modes.

**Internal Data:** In each organization, the client keeps their "**private**" spreadsheets, reports, customer profiles, and sometimes even department databases. This is the internal data, part of which could be useful in a data warehouse.

**Archived Data:** Operational systems are mainly intended to run the current business. In every operational system, we periodically take the old data and store it in achieved files.

**External Data:** Most executives depend on information from external sources for a large percentage of the information they use. They use statistics associating to their industry produced by the external department.

## Data Staging Component

After we have been extracted data from various operational systems and external sources, we have to prepare the files for storing in the data warehouse. The extracted data coming from several different sources need to be changed, converted, and made ready in a format that is relevant to be saved for querying and analysis.

We will now discuss the three primary functions that take place in the staging area.

**1)Data Extraction:** This method has to deal with numerous data sources. We have to employ the appropriate techniques for each data source.

**2)Data Transformation:** As we know, data for a data warehouse comes from many different sources. If data extraction for a data warehouse posture big challenges, data transformation present even significant challenges. We perform several individual tasks as part of data transformation.

First, we clean the data extracted from each source. Cleaning may be the correction of misspellings or may deal with providing default values for missing data elements, or elimination of duplicates when we bring in the same data from various source systems.

Standardization of data components forms a large part of data transformation. Data transformation contains many forms of combining pieces of data from different sources. We combine data from single source record or related data parts from many source records.

On the other hand, data transformation also contains purging source data that is not useful and separating outsource records into new combinations. Sorting and merging of data take place on a large scale in the data staging area. When the data transformation function ends, we have a collection of integrated data that is cleaned, standardized, and summarized.

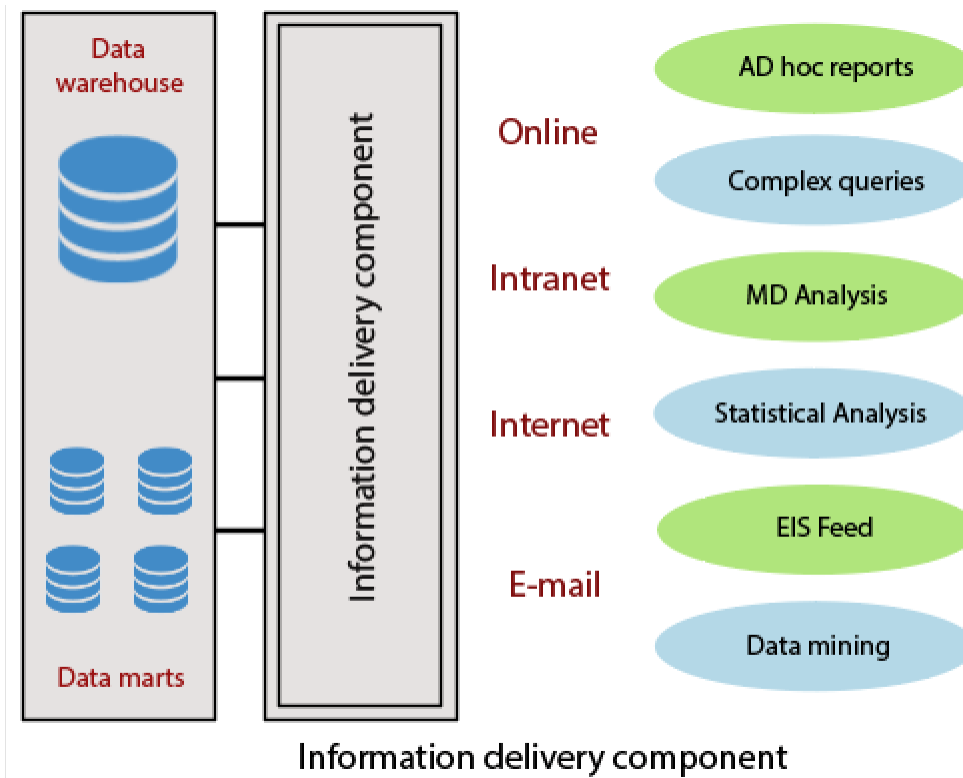
**3)Data Loading:** Two distinct categories of tasks form data loading functions. When we complete the structure and construction of the data warehouse and go live for the first time, we do the initial loading of the information into the data warehouse storage. The initial load moves high volumes of data using up a substantial amount of time.

## Data Storage Components

Data storage for the data warehousing is a split repository. The data repositories for the operational systems generally include only the current data. Also, these data repositories include the data structured in highly normalized for fast and efficient processing.

## Information Delivery Component

The information delivery element is used to enable the process of subscribing for data warehouse files and having it transferred to one or more destinations according to some customer-specified scheduling algorithm.



## Metadata Component

Metadata in a data warehouse is equal to the data dictionary or the data catalog in a database management system. In the data dictionary, we keep the data about the logical data structures, the data about the records and addresses, the information about the indexes, and so on.

## Data Marts

It includes a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to particular selected subjects. Data in a data warehouse should be fairly current, but not mainly up to the minute, although development in the data warehouse industry has made standard and incremental data dumps more achievable. Data marts are lower than data warehouses and usually contain organization. The current trends in data warehousing are to develop a data warehouse with several smaller related data marts for particular kinds of queries and reports.

## Management and Control Component

The management and control elements coordinate the services and functions within the data warehouse. These components control the data transformation and the data transfer into the data warehouse storage. On the other hand, it moderates the data delivery to the clients. It works with the database management systems and authorizes data to be correctly saved in the repositories. It monitors the movement of information into the staging method and from there into the data warehouse storage itself.

## Why we need a separate Data Warehouse?

Data Warehouse queries are complex because they involve the computation of large groups of data at summarized levels.

It may require the use of distinctive data organization, access, and implementation method based on multidimensional views.

Performing OLAP queries in operational database degrade the performance of functional tasks.

Data Warehouse is used for analysis and decision making in which extensive database is required, including historical data, which operational database does not typically maintain.

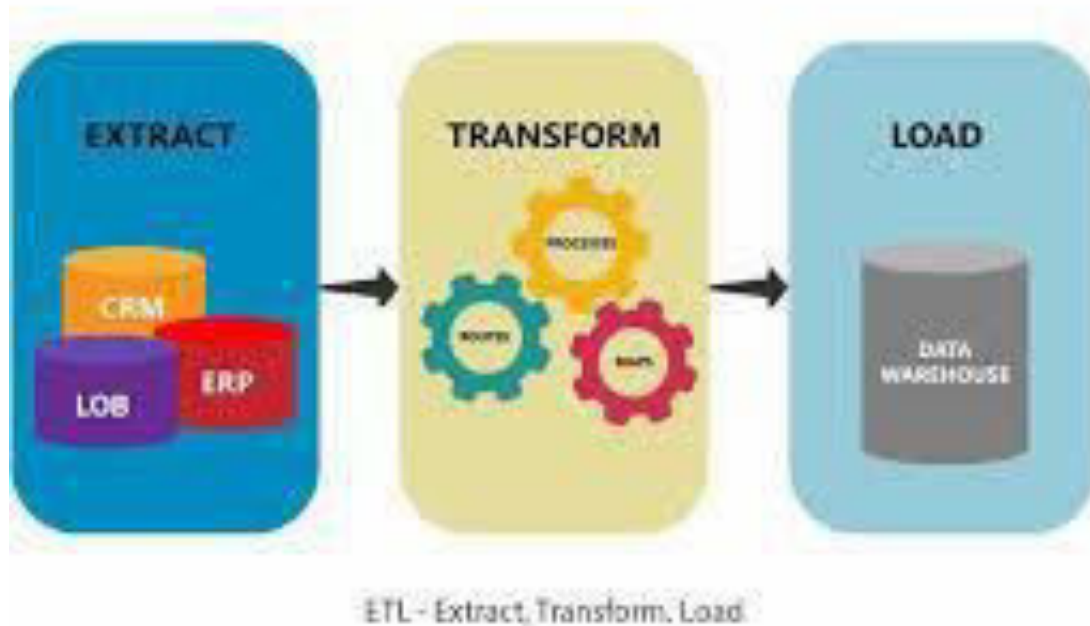
The separation of an operational database from data warehouses is based on the different structures and uses of data in these systems.

Because the two systems provide different functionalities and require different kinds of data, it is necessary to maintain separate databases.

## Difference between Database and Data Warehouse

Database	Data Warehouse
1. It is used for Online Transactional Processing (OLTP) but can be used for other objectives such as Data Warehousing. This records the data from the clients for history.	1. It is used for Online Analytical Processing (OLAP). This reads the historical information for the customers for business decisions.
2. The tables and joins are complicated since they are normalized for RDBMS. This is done to reduce redundant files and to save storage space.	2. The tables and joins are accessible since they are de-normalized. This is done to minimize the response time for analytical queries.
3. Data is dynamic	3. Data is largely static
4. <b>Entity:</b> Relational modeling procedures are used for RDBMS database design.	4. <b>Data:</b> Modeling approach are used for the Data Warehouse design.
5. Optimized for write operations.	5. Optimized for read operations.
6. Performance is low for analysis queries.	6. High performance for analytical queries.
7. The database is the place where the data is taken as a base and managed to get available fast and efficient access.	7. Data Warehouse is the place where the application data is handled for analysis and reporting objectives.

## Extraction transformation load



- **ETL** is a **process** in **Data Warehousing** and it stands for Extract, Transform and Load. It is a **process** in which an **ETL** tool extracts the **data** from various **data** source systems, transforms it in the staging area and then finally, loads it into the **Data Warehouse** system.

## Schema Design

- **Design** a **schema** logically based on business requirements. This can be defined as building a logical model. This **design** step will play an important part in how the data warehouse is developed.

## Star Schema

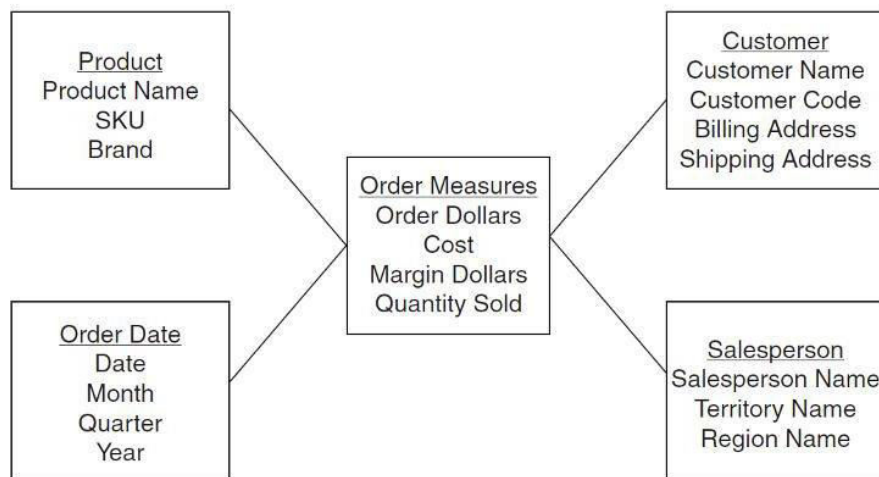


Figure 10-7 Simple STAR schema for orders analysis.

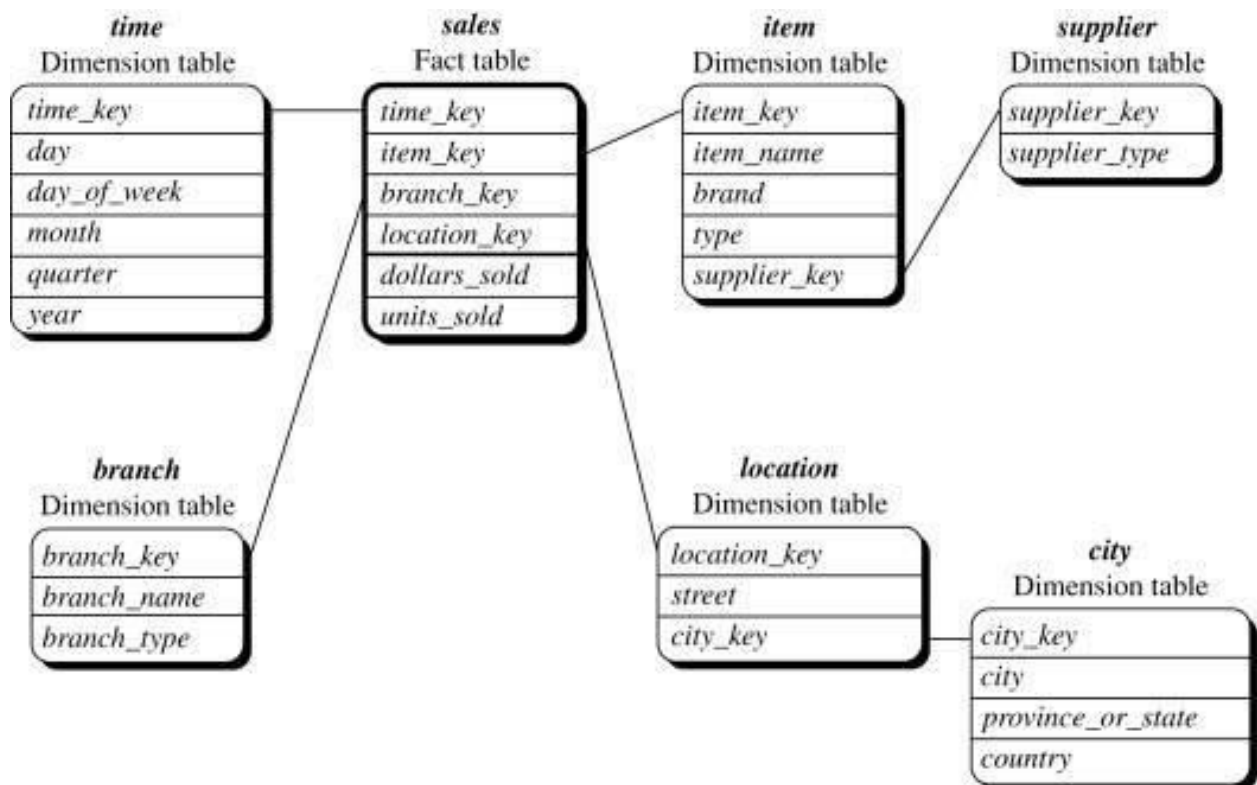
In computing, the **star schema** is the simplest style of data mart **schema** and is the approach most widely used to develop data warehouses and dimensional data marts. The **star schema** consists of one or more fact tables referencing any number of dimension tables.

- It consists of the orders fact table shown in the middle of the schema diagram
- The fact table are the four dimension tables of
- Customer, salesperson, order date and product



- The star schema structure is a structure that can be easily understood and they can comfortably work.
- The **fact table** mainly consists of business facts and foreign keys that refer to primary keys in the **dimension tables**.  
A **dimension table** consists mainly of descriptive attributes that are textual fields. ... On the contrary, a **fact table** contains a foreign key, measurements, and degenerated **dimensions**

## Snow Flake Schema



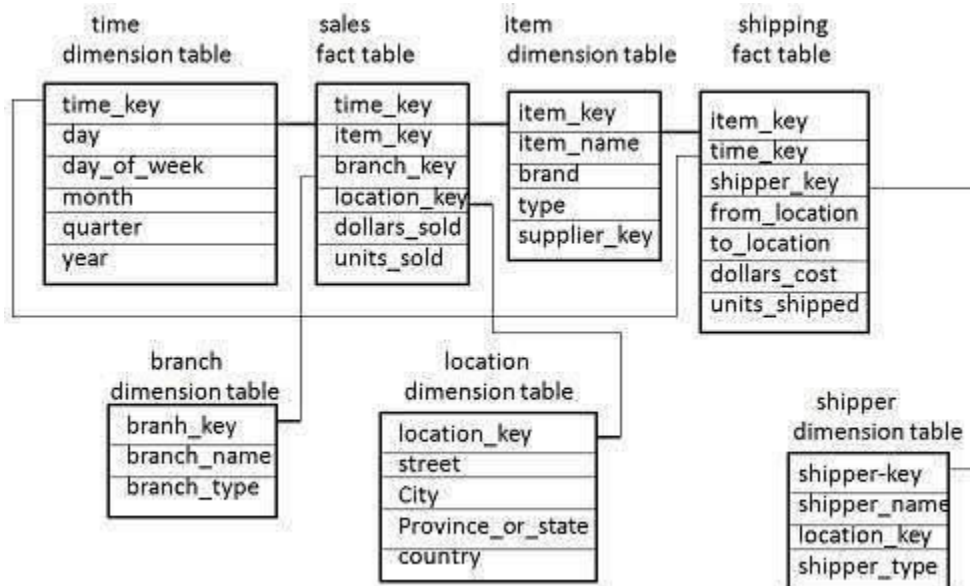
- In **data warehousing**, snowflaking is a form of dimensional modeling in which dimensions are stored in multiple related

dimension tables. A **snowflake schema** is a variation of the star **schema**. ... A star **schema** stores all attributes for a dimension into one denormalized (“flattened”) table.

- In computing, a **snowflake schema** is a logical arrangement of tables in a multidimensional database such that the entity relationship **diagram** resembles a **snowflake** shape.

The **snowflake schema** is represented by centralized fact tables which are connected to multiple dimensions..

### Fact Constellation Schema



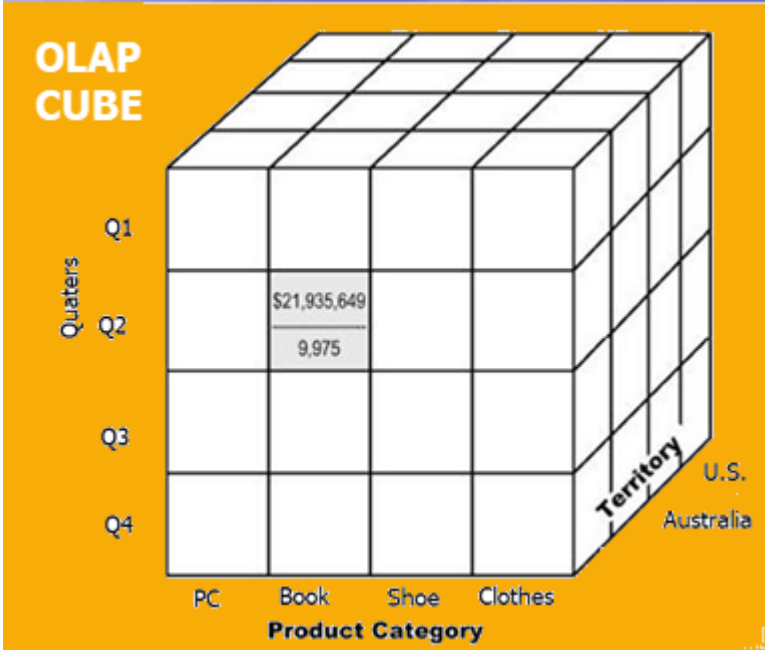
- A **Fact constellation** means two or more **fact** tables sharing **one** or more dimensions. It is also called **Galaxy schema**. ... **Example:** A **fact constellation schema** is shown in the figure below. This **schema** defines two **fact** tables, sales, and shipping.

- **Fact Table.** ... A **fact table** is found at the center of a **star schema** or snowflake **schema** surrounded by dimension **tables**. A **fact table** consists of facts of a particular business process e.g., sales revenue by month by product. Facts are also known as measurements or metrics.

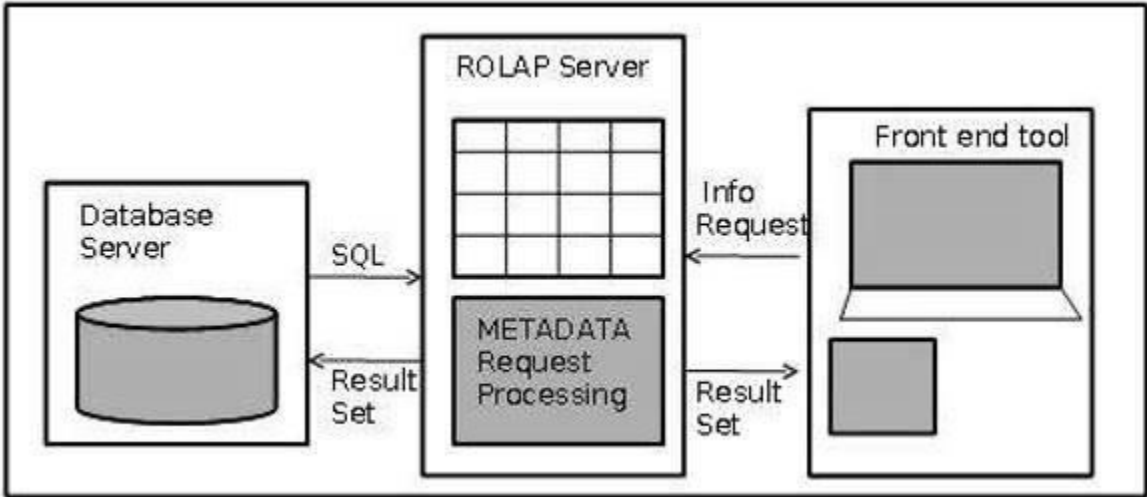
### Non Additive Measures

- fact table is the one which consists of the measurements, metrics or facts of business process. These measurable facts are used to know the business value and to forecast the future business. The different types of facts are explained in detail below:
- Additive measures: These are those specific class of fact measures which can be aggregated across all dimension and their hierarchy. Example: one may tend to add sales across all quarters to avail the yearly sales.
- Semi-Additive measures: These are those specific class of fact measures which can be aggregated across all dimension and their hierarchy except the time dimension. Example: Daily balances fact can be summed up through the customers dimension but not through the time dimension.
- Non-additive measures: These are those specific class of fact measures which cannot be aggregated across all/any dimension and their hierarchy. Example: Facts which have percentages, ratios calculated.

# OLAP Server Architecture

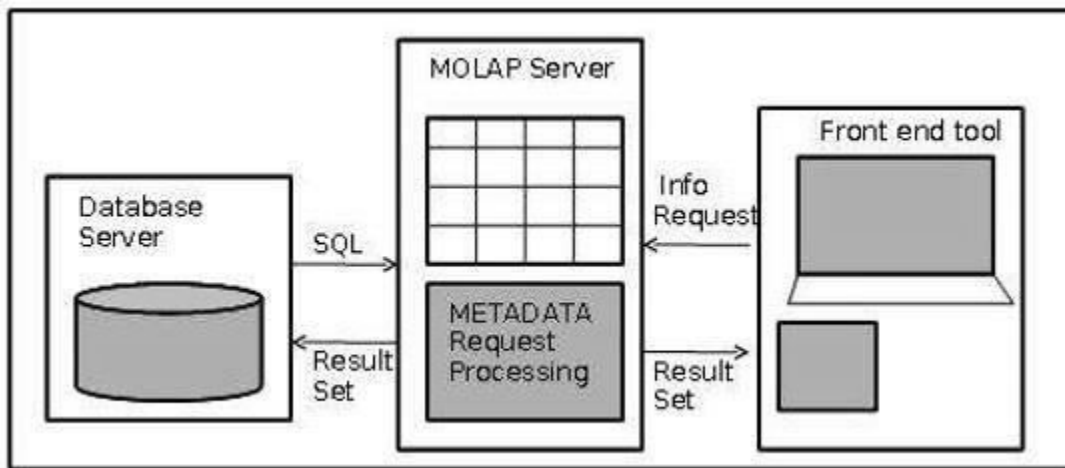


# ROLAP



- Relational online analytical processing (**ROLAP**) is a form of online analytical processing (OLAP) that performs dynamic multidimensional analysis of data stored in a relational database rather than in a multidimensional database (which is usually considered the OLAP standard).
- **OLAP** stores all data, including aggregations, in the source relational database. ... **ROLAP's** advantages include better scalability, enabling it to handle huge amounts of data, and the ability to efficiently manage both numeric and textual data

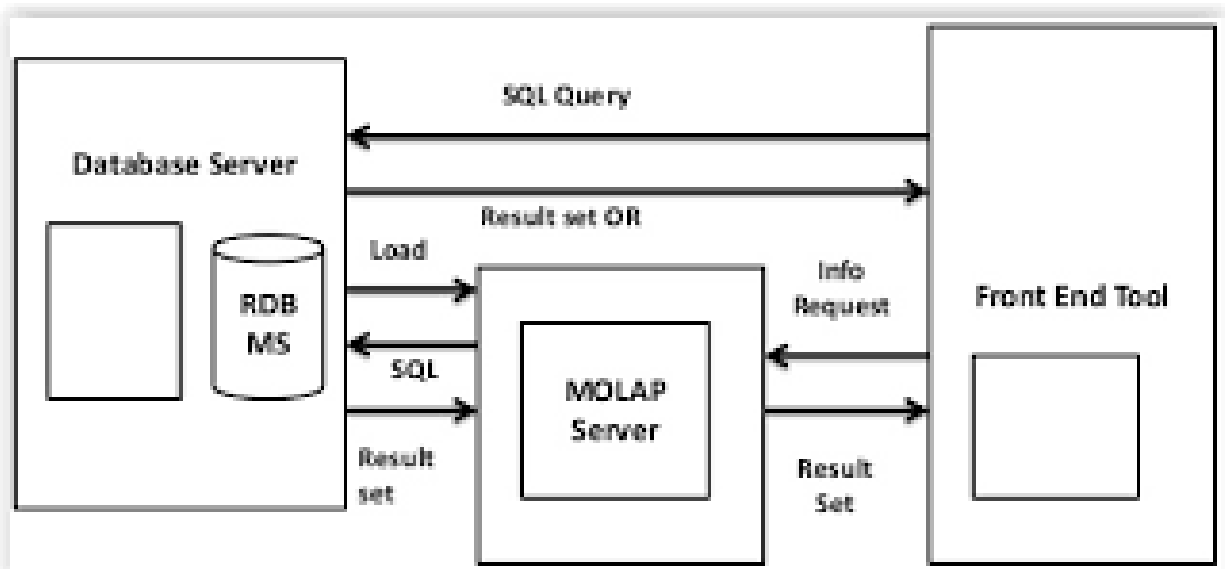
## MOLAP



- Multidimensional online analytical processing (**MOLAP**) is a kind of online analytical processing (OLAP) that, like relational online analytical processing (ROLAP), uses a multidimensional data model to analyze data

- **OLAP** tools enable users to analyze different dimensions of multidimensional data. For example, it provides time series and trend analysis views. **OLAP** often is **used** in data mining. The chief component of **OLAP** is the **OLAP** server, which sits between a client and a database management systems (DBMS).

## HOLAP



- **HOLAP** (Hybrid Online Analytical Processing) is a combination of ROLAP (Relational OLAP) and MOLAP (Multidimensional OLAP). **HOLAP** allows storing part of the data in a MOLAP store and another part of the data in a ROLAP store. **HOLAP** can use varying combinations of ROLAP and OLAP technology.

**Hybrid OLAP** is a mode of storage that uses a combination of multidimensional data structures and relational database tables

for storing multidimensional data. The analysis services stores the aggregations for a HOLAP partition in a multidimensional structure and the facts are stored in relational database

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh52**

**73**

**@gmail.com**



**S.P. Mandali Pune  
Prin. K. P. Mangalvedhekar Institute of  
Management Career Development and  
Research.**

**156-B Railway Lines Solapur.  
Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject Data Mining  
Programme:- BCAIII Sem V**

## Data Mining

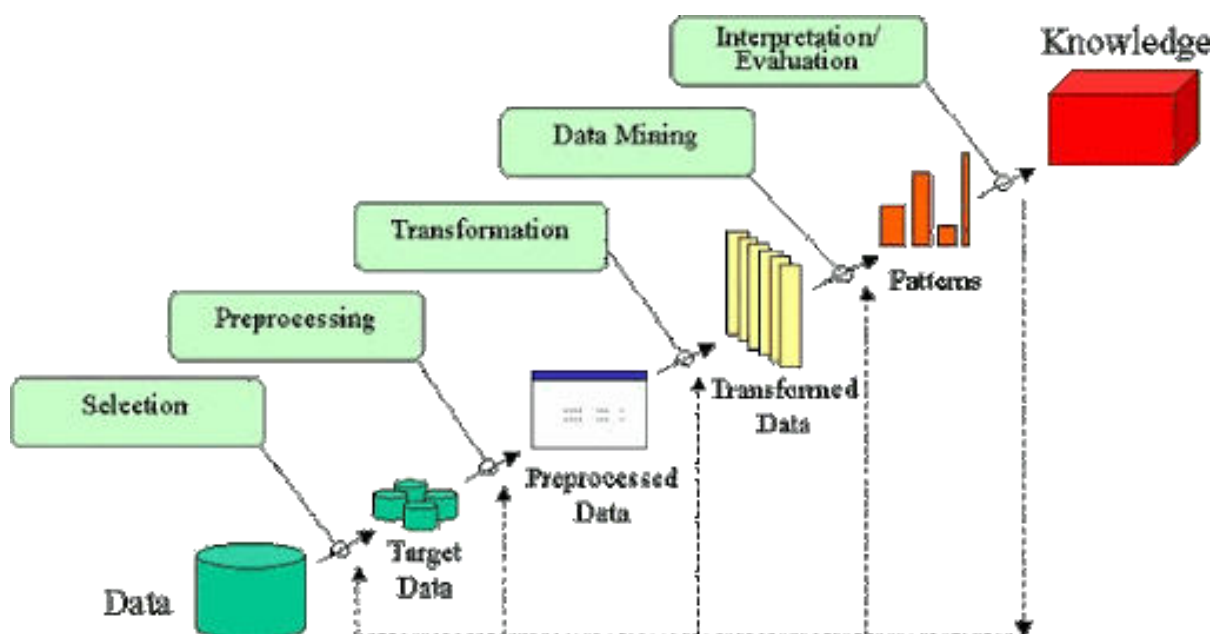
- Introduction to data mining
- **data mining** is defined as a process used to extract usable **data** from a larger set of any raw **data**. It implies analysing **data** patterns in large batches of **data** using one or more software. ... **Data mining** is also known as Knowledge Discovery in **Data** (KDD).

## Difference between Data warehouse and Data Mining

- A **data warehouse** is built to support management functions whereas **data mining** is used to extract useful information and patterns from **data**. **Data warehousing** is the process of compiling information into a **data warehouse**.

## KDD

- The long form of KDD is Knowledge Discovery in databases
- The term Knowledge Discovery in Databases, or **KDD** for short, refers to the broad process of finding knowledge in data, and emphasizes the "high-level" application of particular data mining methods.



- The **KDD process** is an iterative **process** that consists in the selection, cleaning and transformation of data coming not only from databases but also from other heterogeneous sources, such as plain text, data warehouses, images, sound, etc., aimed to apply to them data mining algorithms in order to discover valid, novel,

### Data Mining Tasks

- Now a days, data mining is used in almost all the places where a large amount of data is stored and processed. For example, banks typically use 'data mining' to find out their prospective customers who could be interested in credit cards, personal loans or insurances as well. Since banks have the transaction details and detailed profiles of their customers, they analyze all this data and try to find out patterns which help them predict that certain customers could be interested in personal loans etc.
- **Main Purpose of Data Mining**  
Basically, the information gathered from Data Mining helps to predict hidden patterns, future trends and behaviors and allowing businesses to take decisions.
- Technically, data mining is the computational process of analyzing data from different perspective, dimensions, angles and categorizing/summarizing it into meaningful information.  
Data Mining can be applied to any type of data e.g. Data Warehouses, Transactional Databases, Relational Databases, Multimedia Databases, Spatial Databases, Time-series Databases, World Wide Web.
- **Data Mining as a whole process**  
The whole process of Data Mining comprises of three main phases:
  1. Data Pre-processing – Data cleaning, integration, selection and transformation takes place
  2. Data Extraction – Occurrence of exact data mining
  3. Data Evaluation and Presentation – Analyzing and presenting results
- **Applications of Data Mining**
  1. Financial Analysis
  2. Biological Analysis
  3. Scientific Analysis
  4. Intrusion Detection
  5. Fraud Detection
  6. Research Analysis
- **Real life example of Data Mining – Market Basket Analysis**  
Market Basket Analysis is a technique which gives the careful study of purchases done by a

customer in a super market. The concept is basically applied to identify the items that are bought together by a customer. Say, if a person buys bread, what are the chances that he/she will also purchase butter. This analysis helps in promoting offers and deals by the companies. The same is done with the help of data mining.

## **Steps Involved in Data Preprocessing:**

### **1. Data Cleaning:**

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

- **(a). Missing Data:**

This situation arises when some data is missing in the data. It can be handled in various ways.

Some of them are:

- 1. Ignore the tuples:**

This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

- 2. Fill the Missing values:**

There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

- **(b). Noisy Data:**

Noisy data is a meaningless data that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

- 1. Binning Method:**

This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

- 2. Regression:**

Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

- 3. Clustering:**

This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

### **2. Data Transformation:**

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

- 1. Normalization:**

It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

- 2. Attribute Selection:**

In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

### 3. Discretization:

This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

### 4. Concept Hierarchy Generation:

Here attributes are converted from level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

### 3. Data Reduction:

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

#### 1. Data Cube Aggregation:

Aggregation operation is applied to data for the construction of the data cube.

#### 2. Attribute Subset Selection:

The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p-value of the attribute. The attribute having p-value greater than significance level can be discarded.

#### 3. Numerosity Reduction:

This enables to store the model of data instead of whole data, for example: Regression Models.

#### 4. Dimensionality Reduction:

This reduces the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

Summarisation

- **Summarization** is a key **data mining** concept which involves techniques for finding a compact description of a dataset. Simple **summarization** methods such as tabulating the mean and standard deviations are often applied for exploratory **data** analysis, **data** visualization and automated report generation.

Binaryzation

- **Binarization** is the process of transforming **data** features of any entity into vectors of binary numbers to make classifier algorithms more efficient. In a simple example, transforming an image's gray-scale from the 0-255 spectrum to a 0-1 spectrum is **binarization**.



**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5  
273**

**@gmail.com**

# Subject-Advance Networking

## Design Issues in Network Layer

Network layer is majorly focused on getting packets from the source to the destination, routing error handling and congestion control.

Before learning about design issues in the network layer, let's learn about its various functions.

- **Addressing:**  
Maintains the address at the frame header of both source and destination and performs addressing to detect various devices in network.
- 
- **Packeting:**  
This is performed by Internet Protocol. The network layer converts the packets from its upper layer.
- 
- **Routing:**  
It is the most important functionality. The network layer chooses the most relevant and best path for the data transmission from source to destination.
- 
- **Inter-networking:**  
It works to deliver a logical connection across multiple devices.
- 

### **Network layer design issues:**

The network layer comes with some design issues they are described as follows:

#### **1. Store and Forward packet switching:**

The host sends the packet to the nearest router. This packet is stored there until it has fully arrived once the link is fully processed by verifying the checksum then it is forwarded to the next router till it reaches the destination. This mechanism is called "Store and Forward packet switching."

#### **2. Services provided to Transport Layer:**

Through the network/transport layer interface, the network layer transfers its services to the transport layer. These services are described below.

But before providing these services to the transfer layer following goals must be kept in mind :-

- Offering services must not depend on router technology.
- The transport layer needs to be protected from the type, number and topology of the available router.
- The network addresses for the transport layer should use uniform numbering pattern also at LAN and WAN connections.



Based on the connections there are 2 types of services provided :

- **Connectionless** – The routing and insertion of packets into subnet is done individually. No added setup is required.
- **Connection-Oriented** – Subnet must offer reliable service and all the packets must be transmitted over a single route.

### **3. Implementation of Connectionless Service:**

Packets are termed as “datagrams” and corresponding subnet as “datagram subnets”. When the message size that has to be transmitted is 4 times the size of the packet, then the network layer divides into 4 packets and transmits each packet to router via a few protocols. Each data packet has destination address and is routed independently irrespective of the packets.

### **4. Implementation of Connection Oriented service:**

To use a connection-oriented service, first we establish a connection, use it and then release it. In connection-oriented services, the data packets are delivered to the receiver in the same order in which they have been sent by the sender.

It can be done in either two ways :

- **Circuit Switched Connection** – A dedicated physical path or a circuit is established between the communicating nodes and then data stream is transferred.
- **Virtual Circuit Switched Connection** – The data stream is transferred over a packet switched network, in such a way that it seems to the user that there is a dedicated path from the sender to the receiver. A virtual path is established here. While, other connections may also be using the same path.

**Routing Protocols** are the set of defined rules used by the routers to communicate between source & destination. They do not move the information to the source to a destination, but only update the routing table that contains the information.

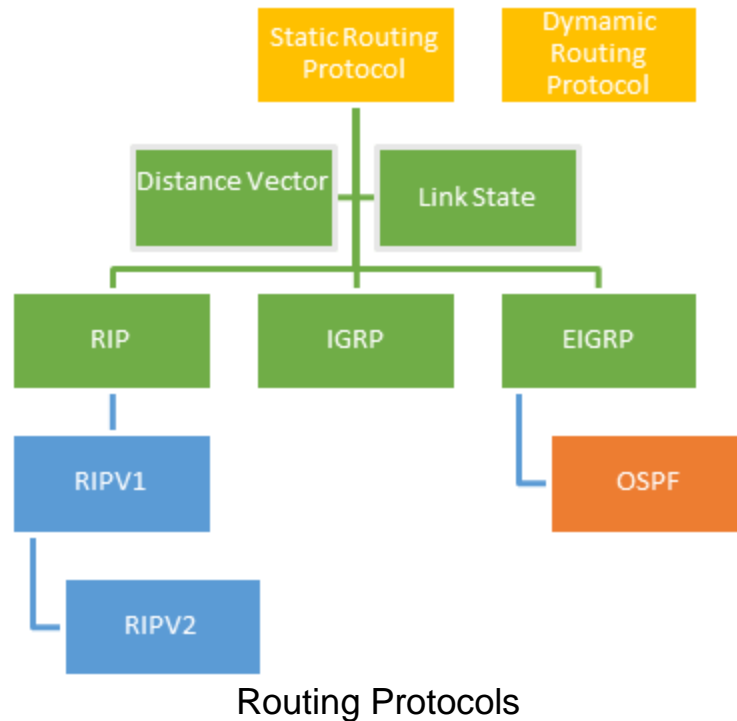
Network Router protocols help you to specify way routers communicate with each other. It allows the network to select routes between any two nodes on a computer network.

## **What You Will Learn**

## **Types of Routing Protocols**

There are mainly two types of Network Routing Protocols

- Static
- Dynamic



## Static Routing Protocols

Static routing protocols are used when an administrator manually assigns the path from source to the destination network. It offers more security to the network.

### Advantages

- No overhead on router CPU.
- No unused bandwidth between links.
- Only the administrator is able to add routes

### Disadvantages

- The administrator must know how each router is connected.
- Not an ideal option for large networks as it is time intensive.
- Whenever link fails all the network goes down which is not feasible in small networks.

# Dynamic Routing Protocols

Dynamic routing protocols are another important type of routing protocol. It helps routers to add information to their routing tables from connected routers automatically. These types of protocols also send out topology updates whenever the network changes' topological structure.

## Advantage:

- Easier to configure even on larger networks.
- It will be dynamically able to choose a different route in case if a link goes down.
- It helps you to do load balancing between multiple links.

## Disadvantage:

- Updates are shared between routers, so it consumes bandwidth.
- Routing protocols put an additional load on router CPU or RAM.

# Distance Vector Routing Protocol (DVR)

Distance Vector Protocols advertise their routing table to every directly connected neighbor at specific time intervals using lots of bandwidths and slow converge.

In the Distance Vector routing protocol, when a route becomes unavailable, all routing tables need to be updated with new information.

## Advantages:

- Updates of the network are exchanged periodically, and it is always broadcast.
- This protocol always trusts route on routing information received from neighbor routers.

## Disadvantages:

- As the routing information are exchanged periodically, unnecessary traffic is generated, which consumes available bandwidth.

## **Internet Routing Protocols:**

The following are types of protocols which help data packets find their way across the Internet:

### **Routing Information Protocol (RIP)**

RIP is used in both LAN and WAN Networks. It also runs on the Application layer of the OSI model. The full form of RIP is the Routing Information Protocol. Two versions of RIP are

1. RIPv1
2. RIPv2

The original version or RIPv1 helps you determine network paths based on the IP destination and the hop count journey. RIPv1 also interacts with the network by broadcasting its IP table to all routers connected with the network.

RIPv2 is a little more sophisticated as it sends its routing table on to a multicast address.

### **Interior Gateway Protocol (IGP)**

IGRP is a subtype of the distance-vector interior gateway protocol developed by CISCO. It is introduced to overcome RIP limitations. The metrics used are load, bandwidth, delay, MTU, and reliability. It is widely used by routers to exchange routing data within an autonomous system.

This type of routing protocol is the best for larger network size as it broadcasts after every 90 seconds, and it has a maximum hop count of 255. It helps you to sustain larger networks compared to RIP. IGRP is also widely used as it is resistant to routing loop because it updates itself automatically when route changes occur within the specific network. It is also given an option to load balance traffic across equal or unequal metric cost paths.

### **Link State Routing Protocol**

Link State Protocols take a unique approach to search the best routing path. In this protocol, the route is calculated based on the speed of the path to the destination and the cost of resources.

## Routing protocol tables:

Link state routing protocol maintains below given three tables:

- **Neighbor table:** This table contains information about the neighbors of the router only. For example, adjacency has been formed.
- **Topology table:** This table stores information about the whole topology. For example, it contains both the best and backup routes to a particular advertised network.
- **Routing table:** This type of table contains all the best routes to the advertised network.

## Advantages:

- This protocol maintains separate tables for both the best route and the backup routes, so it has more knowledge of the inter-network than any other distance vector routing protocol.
- Concept of triggered updates are used, so it does not consume any unnecessary bandwidth.
- Partial updates will be triggered when there is a topology change, so it does not need to update where the whole routing table is exchanged.

## Exterior Gateway Protocol (EGP)

EGP is a protocol used to exchange data between gateway hosts that are neighbors with each other within autonomous systems. This routing protocol offers a forum for routers to share information across different domains. The full form for EGP is the Exterior Gateway Protocol. EGP protocol includes known routers, network addresses, route costs, or neighboring devices.

## Enhanced Interior Gateway Routing Protocol (EIGRP)

EIGRP is a hybrid routing protocol that provides routing protocols, distance vector, and link-state routing protocols. The full form routing protocol EIGRP is Enhanced Interior Gateway Routing Protocol. It will route the same protocols that IGRP routes using the same composite metrics as IGRP, which helps the network select the best path destination.

# Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) protocol is a link-state IGP tailor-made for IP networks using the Shortest Path First (SPF) method.

OSPF routing allows you to maintain databases detailing information about the surrounding topology of the network. It also uses the Dijkstra algorithm (Shortest path algorithm) to recalculate network paths when its topology changes. This protocol is also very secure, as it can authenticate protocol changes to keep data secure.

Distance Vector	Link State
<b>Distance Vector protocol sends the entire routing table.</b>	Link State protocol sends only link information.
<b>It is susceptible to routing loops.</b>	It is less susceptible to routing loops.
Updates are sometimes sent using broadcast.	Uses only multicast method for routing updates.
It is simple to configure.	It is hard to configure this routing protocol.
Does not know network topology.	Know the entire topology.
Example RIP, IGRP.	Examples: OSPF IS-IS.

## **Routing protocols:**

### **Intermediate System-to-Intermediate System (IS-IS)**

ISIS CISCO routing protocol is used on the Internet to send IP routing information. It consists of a range of components, including end systems, intermediate systems, areas, and domains.

The full form of ISIS is Intermediate System-to-Intermediate System. Under the IS-IS protocol, routers are organized into groups called areas. Multiple areas are grouped to make form a domain.

### **Border Gateway Protocol (BGP)**

BGP is the last routing protocol of the Internet, which is classified as a DPVP (distance path vector protocol). The full form of BGP is the Border Gateway Protocol.

This type of routing protocol sends updated router table data when changes are made. Therefore, there is no auto-discovery of topology changes, which means that the user needs to configure BGP manually.

## **What is the purpose of Routing Protocols?**

Routing protocols are required for the following reasons:

- Allows optimal path selection
- Offers loop-free routing
- Fast convergence
- Minimize update traffic
- Easy to configure
- Adapts to changes
- Scales to a large size
- Compatible with existing hosts and routers
- Supports variable length

# Congestion Control techniques in Computer Networks

Congestion control refers to the techniques used to control or prevent congestion. Congestion control techniques can be broadly classified into two categories:

## Open Loop Congestion Control

Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.

### Policies adopted by open loop congestion control –

#### 1. Retransmission Policy :

It is the policy in which retransmission of the packets are taken care of. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.

To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.

#### 2. Window Policy :

The type of window at the sender's side may also affect the congestion. Several packets in the Go-back-n window are re-sent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and make it worse.

Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

#### 3. Discarding Policy :

A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discard the corrupted or less sensitive packages and also be able to maintain the quality of a message.

In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.

#### 4. Acknowledgment Policy :

Since acknowledgements are also the part of the load in the network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.

The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send an acknowledgment



only if it has to send a packet or a timer expires.

### **5. Admission Policy :**

In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

All the above policies are adopted to prevent congestion before it happens in the network.

## **Closed Loop Congestion Control**

Closed loop congestion control techniques are used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:

### **1. Backpressure :**

Backpressure is a technique in which a congested node stops receiving packets from upstream node. This may cause the upstream node or nodes to become congested and reject receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.

In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and inform the source to slow down.

### **2. Choke Packet Technique :**

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets has traveled are not warned about congestion.

### 3. Implicit Signaling :

In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is a congestion.

### 4. Explicit Signaling :

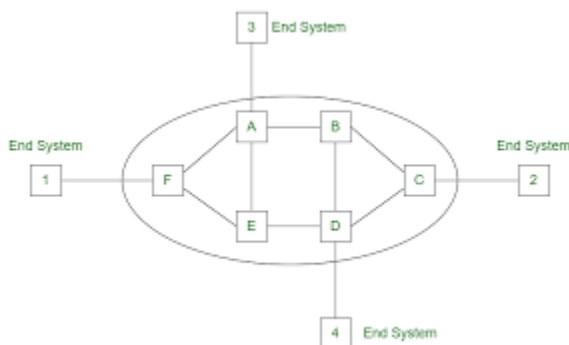
In explicit signaling, if a node experiences congestion it can explicitly send a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating a different packet as in case of choke packet technique.

Explicit signaling can occur in either forward or backward direction.

- **Forward Signaling :** In forward signaling, a signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopts policies to prevent further congestion.
- **Backward Signaling :** In backward signaling, a signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

## Virtual Circuit

**Virtual Circuit** is the computer network providing connection-oriented service. It is a connection-oriented network. In virtual circuit resources are reserved for the time interval of data transmission between two nodes. This network is a highly reliable medium of transfer. Virtual circuits are costly to implement.



### Working of Virtual Circuit:

- In the first step a medium is set up between the two end nodes.
- Resources are reserved for the transmission of packets.
- Then a signal is sent to sender to tell the medium is set up and transmission can be started.
- It ensures the transmission of all packets.
- A global header is used in the first packet of the connection.

- Whenever data is to be transmitted a new connection is set up.

### **Congestion Control in Virtual Circuit:**

Once the congestion is detected in virtual circuit network, closed-loop techniques is used. There are different approaches in this technique:

- **No new connection –**  
No new connections are established when the congestion is detected. This approach is used in telephone networks where no new calls are established when the exchange is overloaded.
- **Participation of congested router invalid –**  
Another approach to control congestion is allow all new connections but route these new connections in such a way that congested router is not part of this route.
- **Negotiation –**  
To negotiate different parameters between sender and receiver of the network, when the connection is established. During the set up time, host specifies the shape and volume of the traffic, quality of service and other parameters.

### **Advantages of Virtual Circuit:**

1. Packets are delivered to the receiver in the same order sent by the sender.
2. Virtual circuit is a reliable network circuit.
3. There is no need for overhead in each packet.
4. Single global packet overhead is used in virtual circuit.

### **Disadvantages of Virtual Circuit:**

1. Virtual circuit is costly to implement.
2. It provides only connection-oriented service.
3. Always a new connection set up is required for transmission.

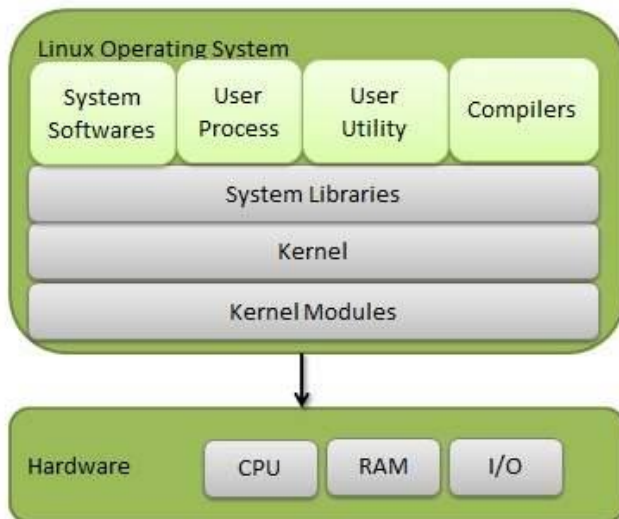
# Linux Operating System

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

## Components of Linux System

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** – System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks.



## Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

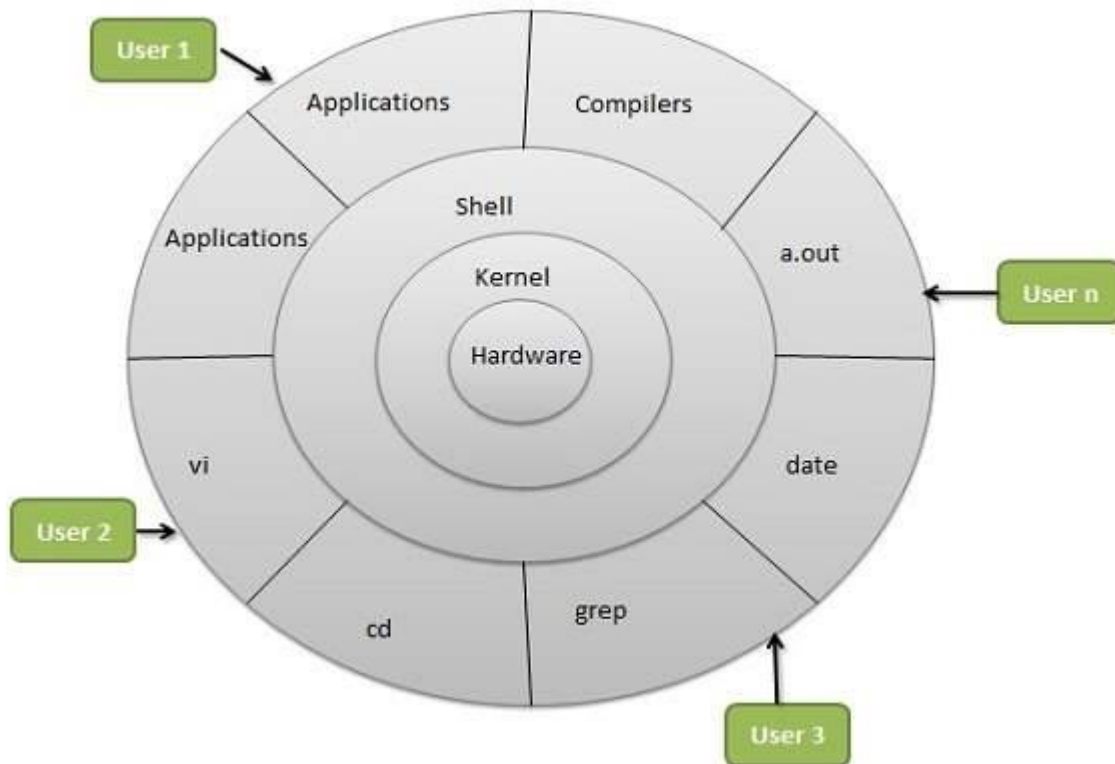
## Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** – Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** – Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** – Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** – Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

## Architecture

The following illustration shows the architecture of a Linux system –



The architecture of a Linux System consists of the following layers –

- **Hardware layer** – Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** – It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** – An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** – Utility programs that provide the user most of the functionalities of an operating systems.

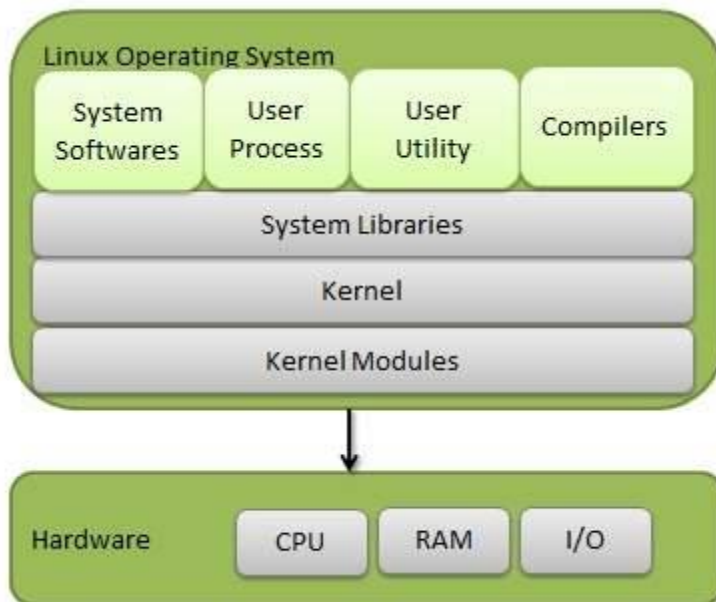
Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

## Components of Linux System

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

- **System Library** – System libraries are special functions or programs using which application programs or system utilities access Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks.



## Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

## Basic Features

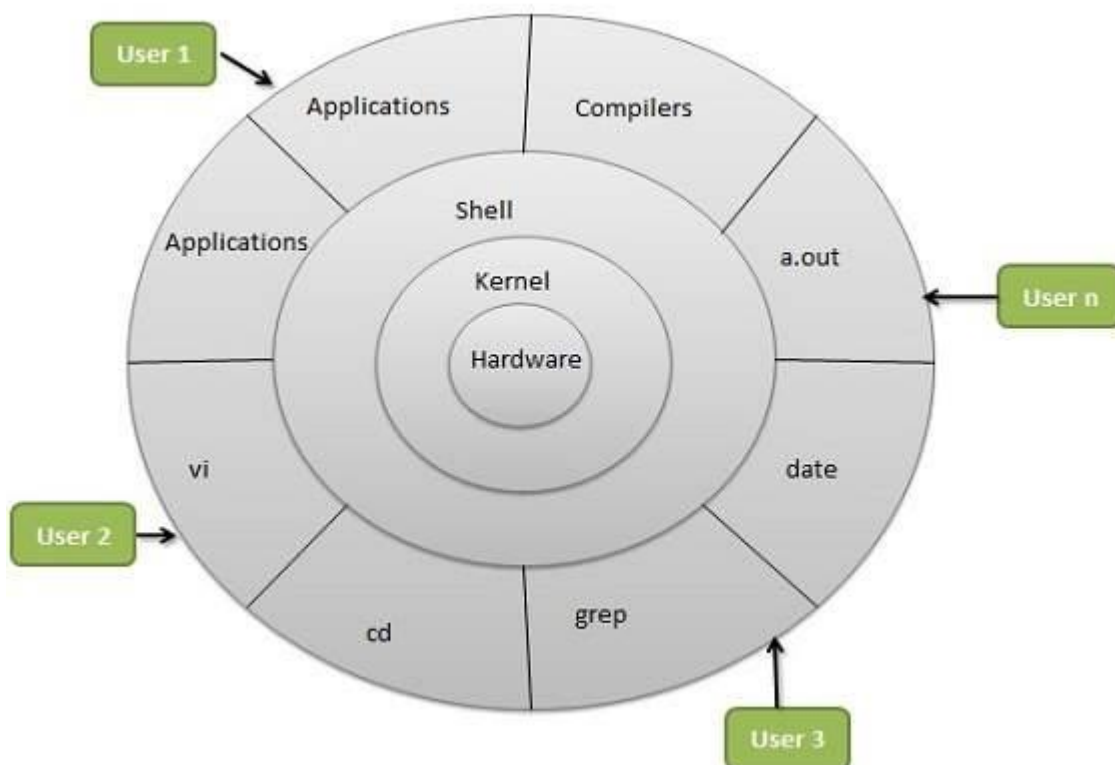
Following are some of the important features of Linux Operating System.

- **Portable** – Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** – Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

- **Multi-User** – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** – Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** – Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

## Architecture

The following illustration shows the architecture of a Linux system –



The architecture of a Linux System consists of the following layers –

- **Hardware layer** – Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).



- **Kernel** – It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** – An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** – Utility programs that provide the user most of the functionalities of an operating systems.

## History-

In this article, you will learn what Linux is, where Linux came from, what a Linux distribution is and some of the reasons that Linux is favored for certain applications and projects over other operating systems.

First and foremost, Linux is an operating system. An operating system is simply a collection of software that manages hardware resources and provides an environment where applications can run. The operating system allows applications to store information, send documents to printers, interact with users and other things.

Linux is also a kernel. Typically, when the term “Linux” is used, it refers to the Linux operating system as a whole. However, it can refer to just the Linux kernel as well. The Linux kernel is the core or the heart of the operating system. It's the layer that sits between the hardware and applications. Said another way, it's the intermediary between software and hardware. However, to have a useful operating system, you need other components in addition to the kernel. These components can include system libraries, graphical user interfaces, email utilities, web browsers and other programs.

Linus Torvalds created Linux when he was a student at the University of Helsinki studying computer science. In early 1991 he purchased an IBM-compatible personal computer that came with the MS-DOS operating system. Linus wasn't satisfied with MS-DOS and wanted to use a UNIX operating system like he was accustomed to at the University. When he set out to obtain a copy of UNIX for his personal use, he found that the least expensive UNIX he could buy was about \$5,000 USD. Driven by the desire to run a UNIX-like operating system on his personal computer, he set out to create Linux. Linus and over 100 developers worked on Linux over the next couple of years and in March of 1994, version 1.0 of the Linux kernel was released.

Linux is open source software. This means that anyone can use, copy, study and change the software in any way they chose so long as the source code is openly shared with others. To date, thousands of people have made improvements to Linux. With Linux being free and open source software, it has led to the rise of Linux distributions. In every case, the source code is free, but in some cases, the distribution is not free – the binaries, the compiled code is not free. For example, you have to pay a license in order to run Red Hat Enterprise Linux. However, Red Hat releases their source code for anyone to download.

Again, Linux is not a UNIX-derivative. It was written from scratch. However, many of the commands that are found in Linux are also found in UNIX. If you have any experience on UNIX systems, you're going to feel right at home on a Linux system.

A Linux distribution is the Linux kernel and a collection of software that together, create an operating system. Each distribution has its own goals and areas of focus. Your choice of distribution will depend on what you're trying to accomplish. There are distributions that are commercial. These commercial Linux distributions are backed by corporations and you can buy support from them. There are non-commercial Linux distributions. These are maintained by a community of volunteers. You have Linux distributions that are designed for server use, others that are designed for desktop use, some that are focus on research and science. There are others that are focused on multimedia production. There are literally hundreds of Linux distributions.

DistroWatch.com is a great website to learn about all the available Linux distributions. "Distro" is short for distribution. Here are some popular Linux distributions:

- RedHat Enterprise Linux (RHEL)
- Fedora
- Ubuntu
- Debian
- SuSE Linux Enterprise Server (SLES)
- OpenSuSE
- Linux Mint

Again, there are a hundred of Linux distros. These are just a few of the most popular Linux distributions. To get an idea of exactly what's available, go to DistroWatch.com.

So what are some of the reasons you would want to run Linux?

Linux runs on many hardware platforms from dedicated networking devices to phones to personal computers and even super computers. Proprietary UNIX operating systems typically only run on their hardware from their company. For example, HPUX only runs on HP servers, AIX only runs on IBM servers. Linux can run on HP, IBM and other servers. Linux was developed on PC hardware using Intel processors. Over time, Linux has been ported to more hardware platforms than any other operating system.

The small footprint of Linux allows it to run on older hardware or on embedded systems. Also, Linux is known for being stable, reliable and secure. This makes it a great source for servers that need to continuously run without downtime.

Linux has traditionally been used for server applications. Linux can be used to host websites, act as file servers and even run database software.

Linux doesn't have to be used as a server though. Many people find Linux to be a good everyday operating system that they use on their personal desktops.

Linux is free. Not only is the source code freely available, but you can run Linux on your hardware without having to pay a licensing fee in many cases. However, if your business depends on servers that are running Linux, having a commercial Linux distribution which you pay for and having someone that can provide support can be well worth your while.

Linux is typically not as costly as the proprietary UNIX operating systems. Linux is also free in the sense that you can use it for any purpose and you can modify it to fit your needs if you so desire. There are also many free software applications that run on Linux. Many of these free applications were written with Linux in mind.

Let's recap what we've gone over in this article. First, we learned that Linux is an operating system. When someone says the word "Linux" without a qualifier, more than likely they are talking about the entire operating system. However, Linux is also a kernel. When people speak of the Linux kernel, they are typically specific and say "the Linux kernel."

Linux, being the operating system, is the intermediary between hardware and software. Linux distributions are implementations of Linux. Each Linux distribution has a different goal and a slightly different focus. Your choice of distribution will be driven by the goal you are trying to accomplish.

- 

In this chapter, we will discuss in detail about file management in Unix. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

When you work with Unix, one way or another, you spend most of your time working with files. This tutorial will help you understand how to create and remove files, copy and rename them, create links to them, etc.

In Linux, there are so many choices, and this includes the desktop environments and window managers. The most popular [desktop environments](#) in Linux are [GNOME](#), [Unity](#), [Cinnamon](#), [MATE](#), [KDE](#), [Xfce](#), and [LXDE](#). All of them offer sophisticated point-and-click [graphical user interfaces \(GUI\)](#) which are on par with the desktop environments found in [Windows](#) and [Mac OS X](#). When you ask different people which of these are the best, you will likely get many different answers. So which is the best? Well..... it is largely a matter of opinion, and the

capabilities of your computer hardware can also be important in deciding. For example, users with older computers will be better served to choose Xfce or especially LXDE, while users with newer hardware can get more desktop effects by choosing KDE, Cinnamon, or GNOME. Another consideration when choosing a desktop environment is your preference for customizing it. If you like to have a lot of options to customize and tweak your desktop, then KDE will by default give you the greatest flexibility to do this. Xfce comes next, and then LXDE, while Unity and the default GNOME 3.x shell offer relatively few options in the way of desktop customization. Personally, I like all of them, and if you have the time and are a bit adventurous, then I recommend you try each of the major desktop environments described below, as well as others such as [Enlightenment](#) and [Razor-qt](#) and decide which of them works best for you. GNOME, Unity, Cinnamon, MATE, KDE, Xfce, LXDE, Enlightenment, and Razor-qt are all excellent and are definitely worth consideration.

---











## A BRIEF DESCRIPTION OF GNOME, UNITY, CINNAMON, MATE, KDE, XFCE, LXDE,

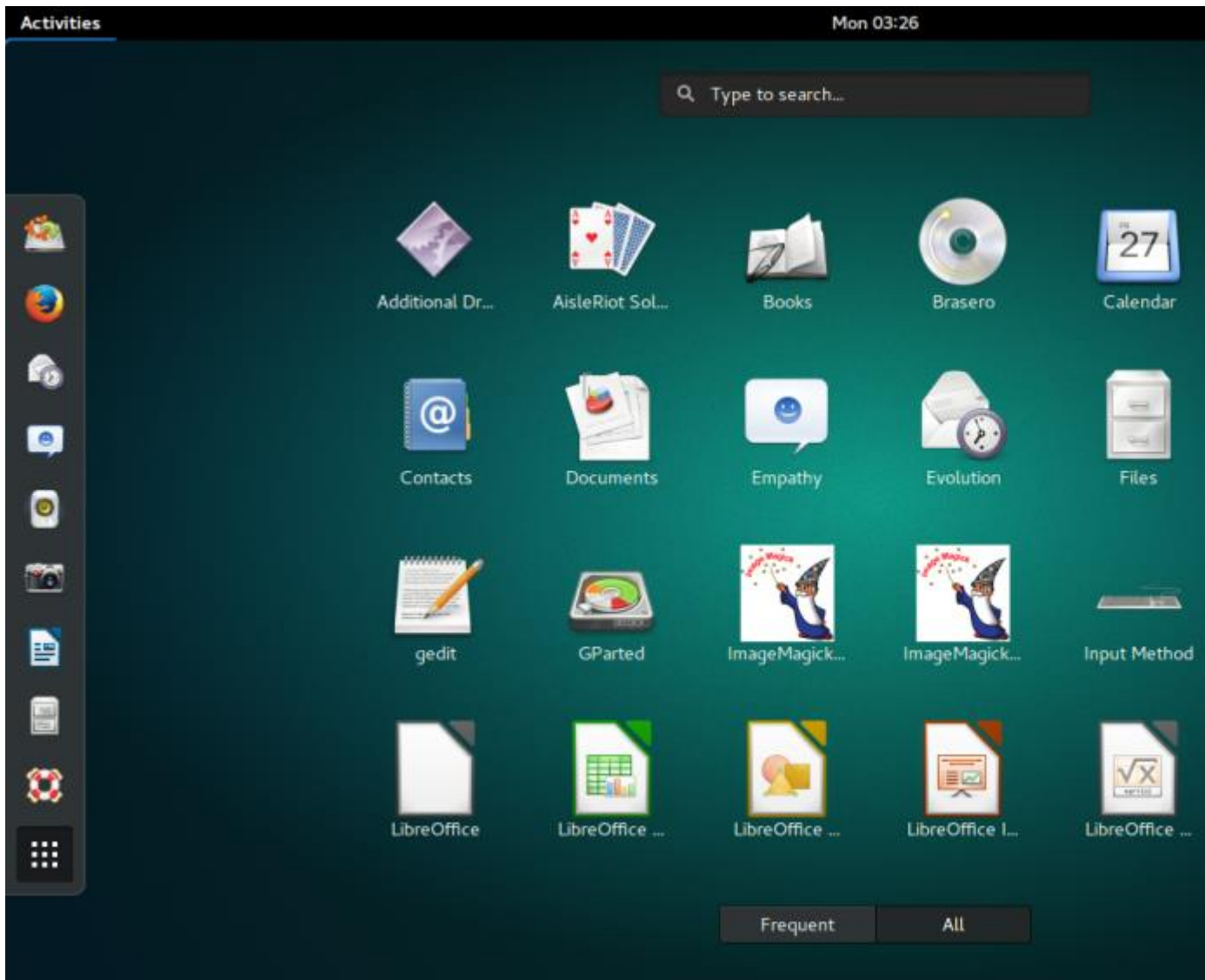
---

### [GNOME](#)

The GNOME ([GNU](#) Network Object Model Environment) has been redesigned with the most recent GNOME 3.x and is a wide departure from the traditional GNOME 2.x desktop. The newly released GNOME 3.x with its GNOME Shell user interface is a drastic change from the "classic" GNOME 2.x shell. While the GNOME 3.x shell is fairly intuitive, for someone who is accustomed to GNOME 2.x, or any other desktop environment for that matter, there will be a considerable amount of adjustment. In the GNOME 3.x shell, there is only one [panel](#) located at the top of the desktop, and there is no longer a traditional menu. To open programs, users can either press the Windows key, or they can click on "Activities" found on the left side of the panel. This gives the options of a program launcher that appears on the left side of the desktop, an "Applications" option found on the upper left part of the desktop (which is the closest thing to a menu), or they can search for programs using the search box on the upper right of the desktop. Additionally, when clicking on "Activities," a desktop switcher appears on the right side of the desktop. Another change involves the buttons on the windows; in GNOME 2.x, and practically every other desktop environment or window manager, there are at least three buttons found at the top of each window: one to exit the window, one to maximize the window, and one to minimize the window. However, in the default GNOME 3.x shell, there is only one button that is used to exit the window, which really takes some getting used to. If you want your laptop or desktop to look and behave like a cell phone or tablet, then the GNOME 3.x shell might be for you. Overall, the GNOME 3.x shell is a very simple, clean, and visually pleasing desktop. [Debian](#), and [Fedora](#) are major distros which use some form of GNOME in their main editions.

🔍 Type to search...

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 



It should also be mentioned that GNOME has a wealth of applications which are designed for its desktop, but they can also be used in the other desktop environments as well; [click here](#) to see a list of them [1]. Following are a few applications and components of GNOME:

Window Manager: [Mutter](#) (GNOME 3.x shell)

File Manager: [Nautilus](#)

Office Suite: [GNOME Office](#) (which includes [AbiWord](#) and [Gnumeric](#))

Music Player: [Rhythmbox](#)

Video Player: [Totem](#)

CD/DVD Burner: [Brasero](#)

Games: [GnomeGames](#)

Widget Toolkit: [GTK+](#)

Recommended System Requirements for the GNOME 3.x shell in its default mode

Required RAM	768 MB
Required CPU	400 MHz

---

## KDE

In many ways, KDE (K Desktop Environment) in its default configuration is very similar in appearance to Microsoft Windows and Windows users will likely feel very much at home when using it. KDE is arguably the most powerful, versatile, smoothly integrated, and visually pleasing of all the Linux desktops and has more point-and-click customization options and “eye candy” than any of the various GNOMEs, Xfce, LXDE or any other Linux desktop. With its [Plasma Workspaces](#), users can easily add a variety of [widgets](#) to the desktop. While KDE is the most polished in appearance when compared to other Linux desktops, it can be quite resource-hungry with its many desktop effects. However, when the desktop effects are turned off, KDE is fairly energy efficient. Typically, KDE requires less CPU resources than Ubuntu’s Unity and less RAM than the GNOME 3.x shell. [OpenSUSE](#), [PCLinuxOS](#), [SolydXK](#), [Mageia](#), and [Chakra](#) are some major Linux distros running KDE in their main editions. [Kubuntu](#) is the KDE version of Ubuntu. In summary, KDE is an outstanding desktop environment that is most definitely worth consideration. Pictured above is KDE PCLinuxOS.

Like GNOME, KDE includes a large number of applications which are designed to be used in its desktop, many of which have a name that begins with the letter “K.” For example, [Konqueror](#) is a web browser and file manager, and [KStars](#) is a desktop planetarium. Also like the GNOME applications, the KDE applications can be used in other desktop environments. You can [click here](#) to see a list of KDE applications [2]. Following are a few applications and components of KDE:

Window Manager: [KWin](#)

File Manager: [Dolphin](#)

Office Suite: [KOffice](#)

Music Player: [Amarok](#)

Video Player: [Dragon Player](#)

CD/DVD Burner: [K3b](#)

Terminal Emulator: [Konsole](#)

Games: [The KDE Games Center](#)

Education: [KDE Edu](#)

Widget Toolkit: [Qt](#)

Recommended System Requirements for KDE

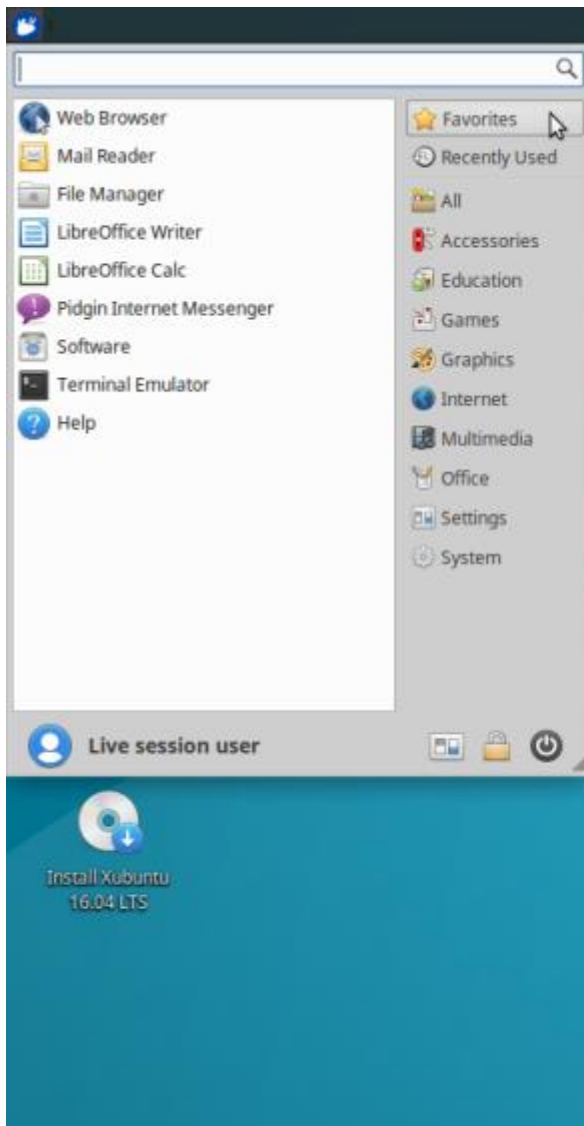
Required RAM	615 MB
Required CPU	1 GHz

One option in newer versions of KDE is to run it in the “Search and Launch” mode, which in some ways is similar in appearance to the GNOME 3.x shell and to Ubuntu’s Unity. This mode with its large icons and search can be used with a touchscreen, and is great for smaller devices such as [netbooks](#) and [tablets](#). While the Search and Launch mode is an option in KDE, it is not the default like it is in the GNOME 3.x shell and in Unity. The Search and Launch mode is easily activated or deactivated by clicking on the “Show Activity Manager” button found



on the desktop panel, next to the "Application Launcher Menu." Click on the picture above to see a larger screenshot of the KDE 4.7 Search and Launch mode.

---



### XFCE

is not same Less resource-hungry than GNOME or KDE, Xfce is a great choice for older computers and it is still a full-fledged desktop environment that offers a great deal to the user. In my opinion, Xfce provides a nice balance between functionality and conservation of system resources, while still having a beautiful desktop. In its default appearance, Xfce very much resembles Mac OS X with its dock-like panel found at the bottom of the desktop. Users can drag their favorite applications from the menu (found on the left side of the upper panel) and place them on the bottom dock/panel in a similar manner as can be done in Mac OS X. Just like GNOME 2.x and KDE, Xfce may easily be customized to more closely resemble Windows, or to be configured



otherwise as desired. SolydX is the Xfce version of [SolydXK](#). Also, [Xubuntu](#) is the Xfce version of Ubuntu, and [Mythbuntu](#) has Xfce as its desktop. [VectorLinux](#) uses Xfce as its default desktop, and many other Linux distros offer Xfce versions as well. In many ways, Xfce looks and acts much like GNOME 2.x, and for those who like the GNOME 2.x desktop and are not completely satisfied with the changes in the GNOME 3.x shell or Unity, Xfce could be a great fit. Pictured above is Xfce in Xubuntu 16.04 LTS.

Following are a few applications and components of Xfce:

Window Manager: [Xfwm](#)

File Manager: [Thunar](#)

Media Player: [Parole](#)

CD/DVD Burner: [Xfburn](#)

Task Manager: [Xfce Task Manager](#)

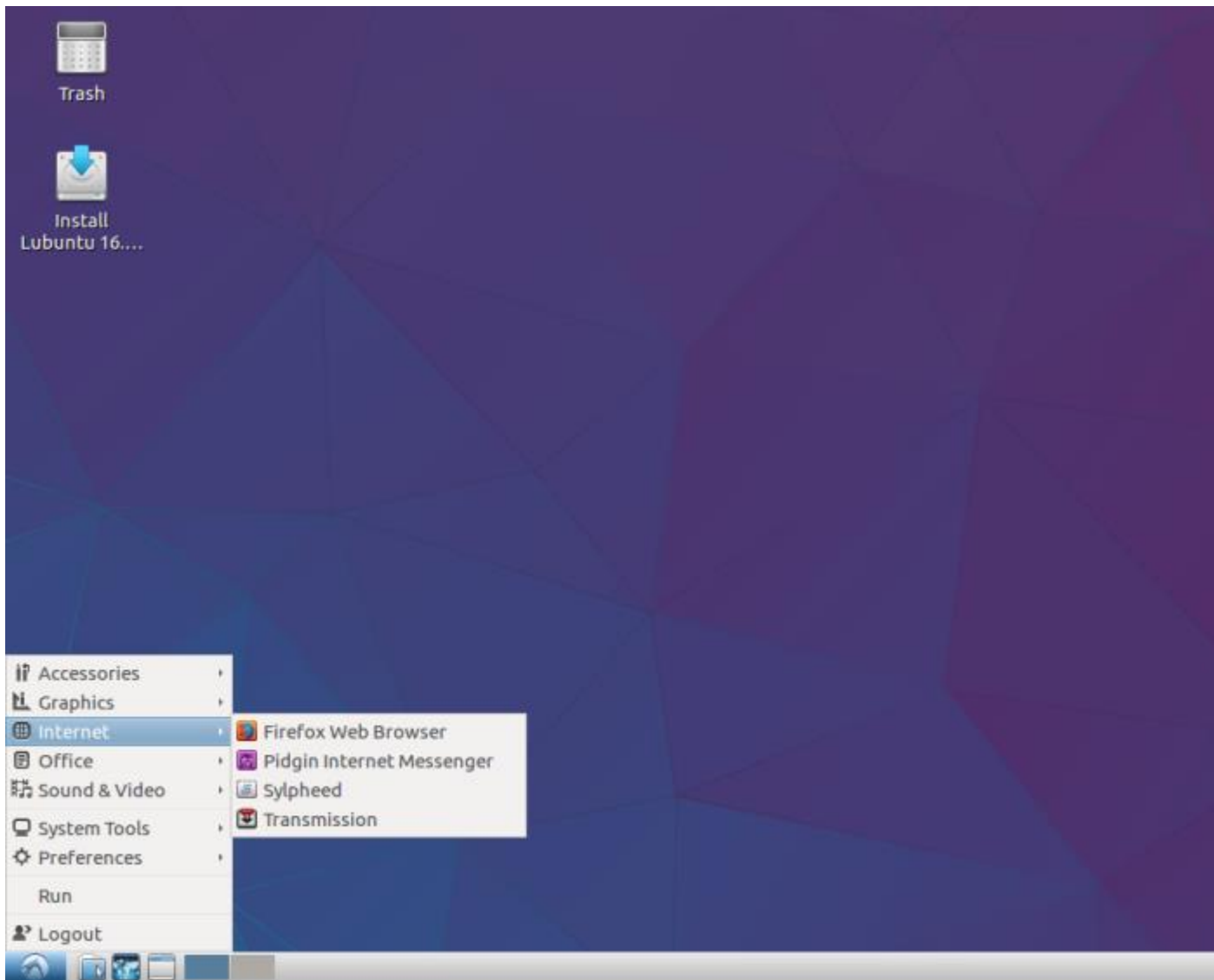
Widget Toolkit: GTK+

Recommended System Requirements for Xfce

Required RAM	192 MB
Required CPU	300 MHz

[http://wiki.xfce.org/minimum\\_requirements](http://wiki.xfce.org/minimum_requirements) [3]

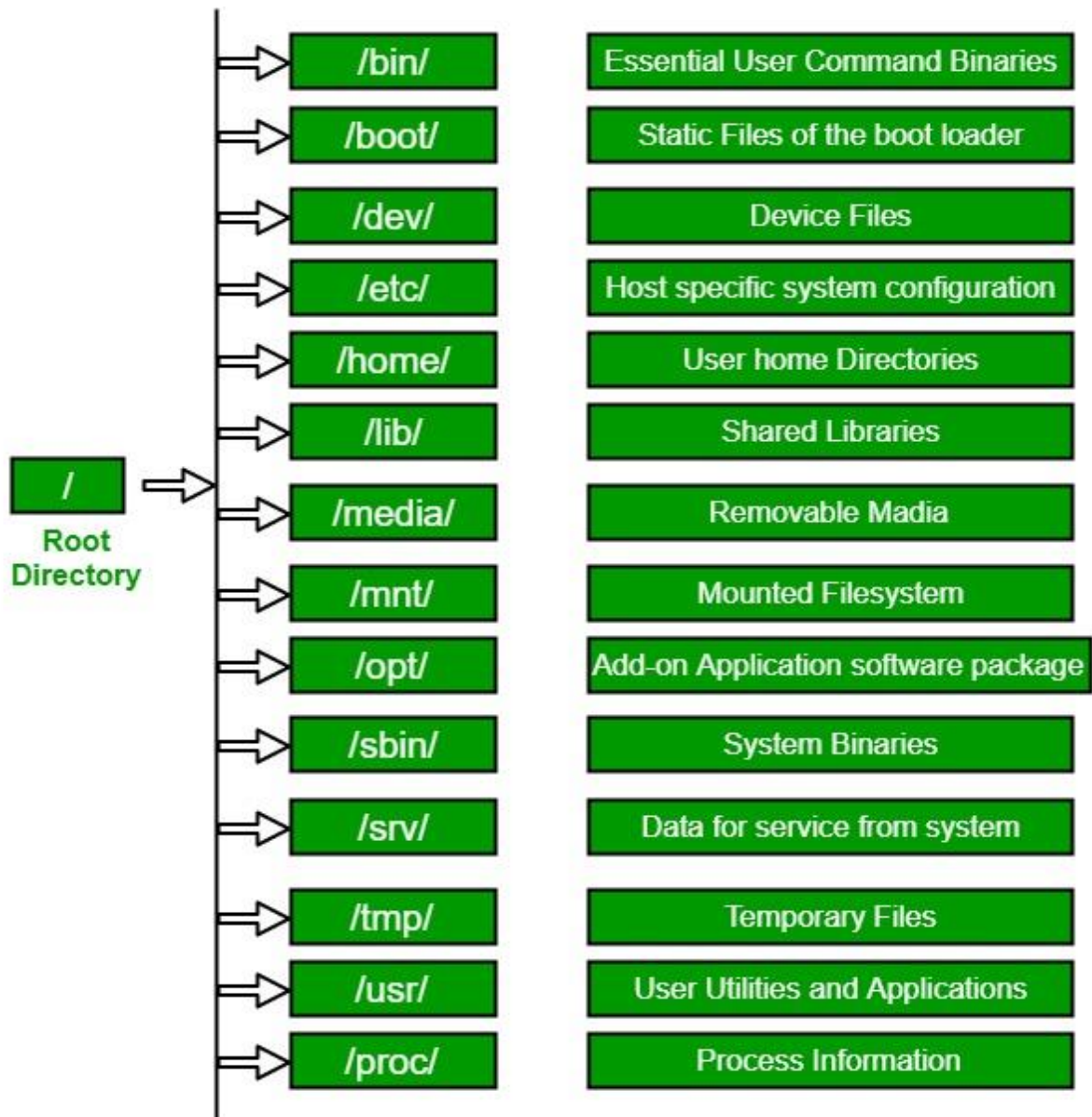
---



## Linux File Hierarchy Structure

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux.



**1. / (Root) :** Primary hierarchy root and root directory of the entire file system hierarchy.

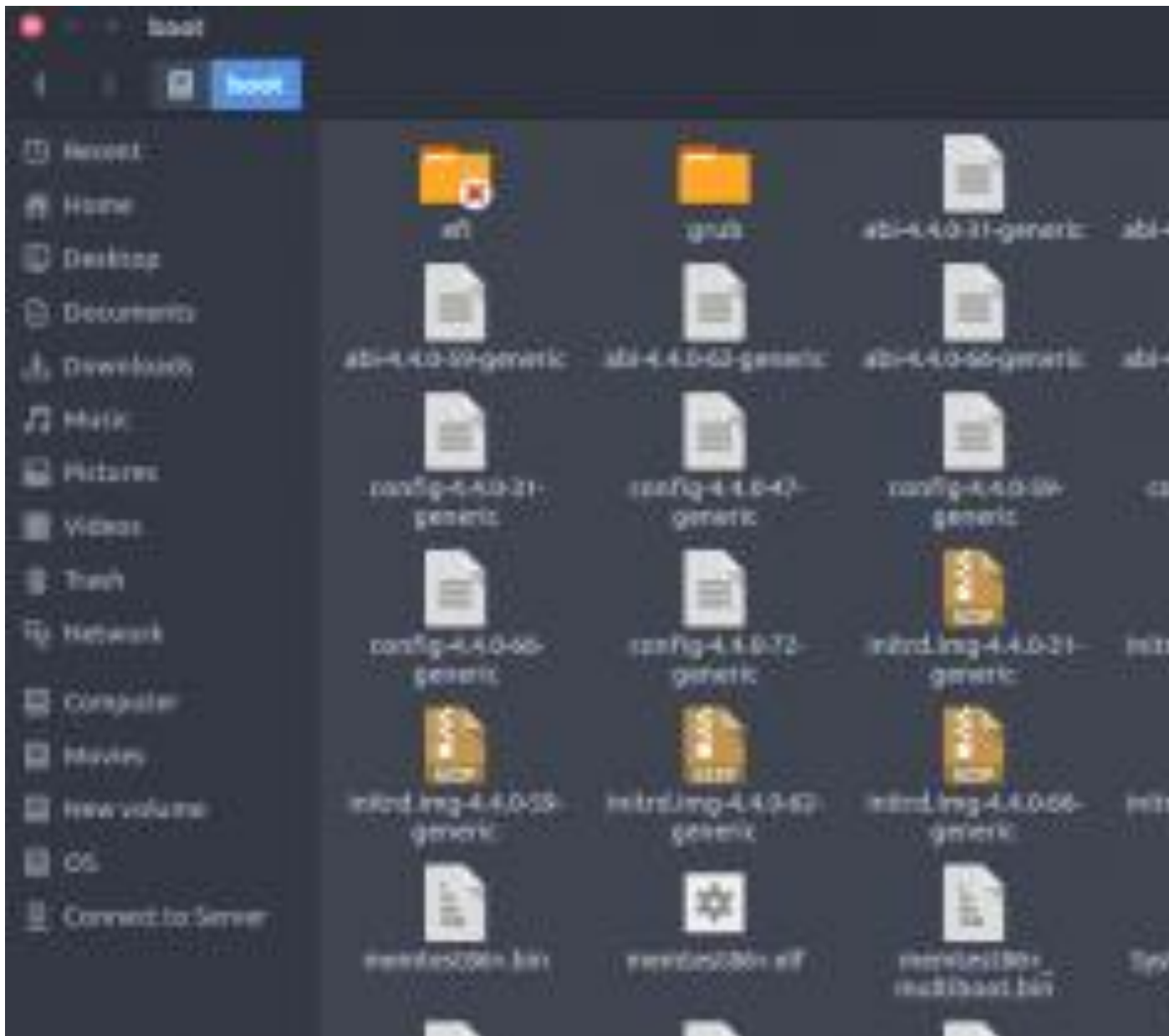
- Every single file and directory starts from the root directory
- Only root user has the right to write under this directory
- /root is root user's home directory, which has /



2. **/bin** : Essential command binaries that need to be available in single user mode; for all users, e.g., cat, ls, cp.

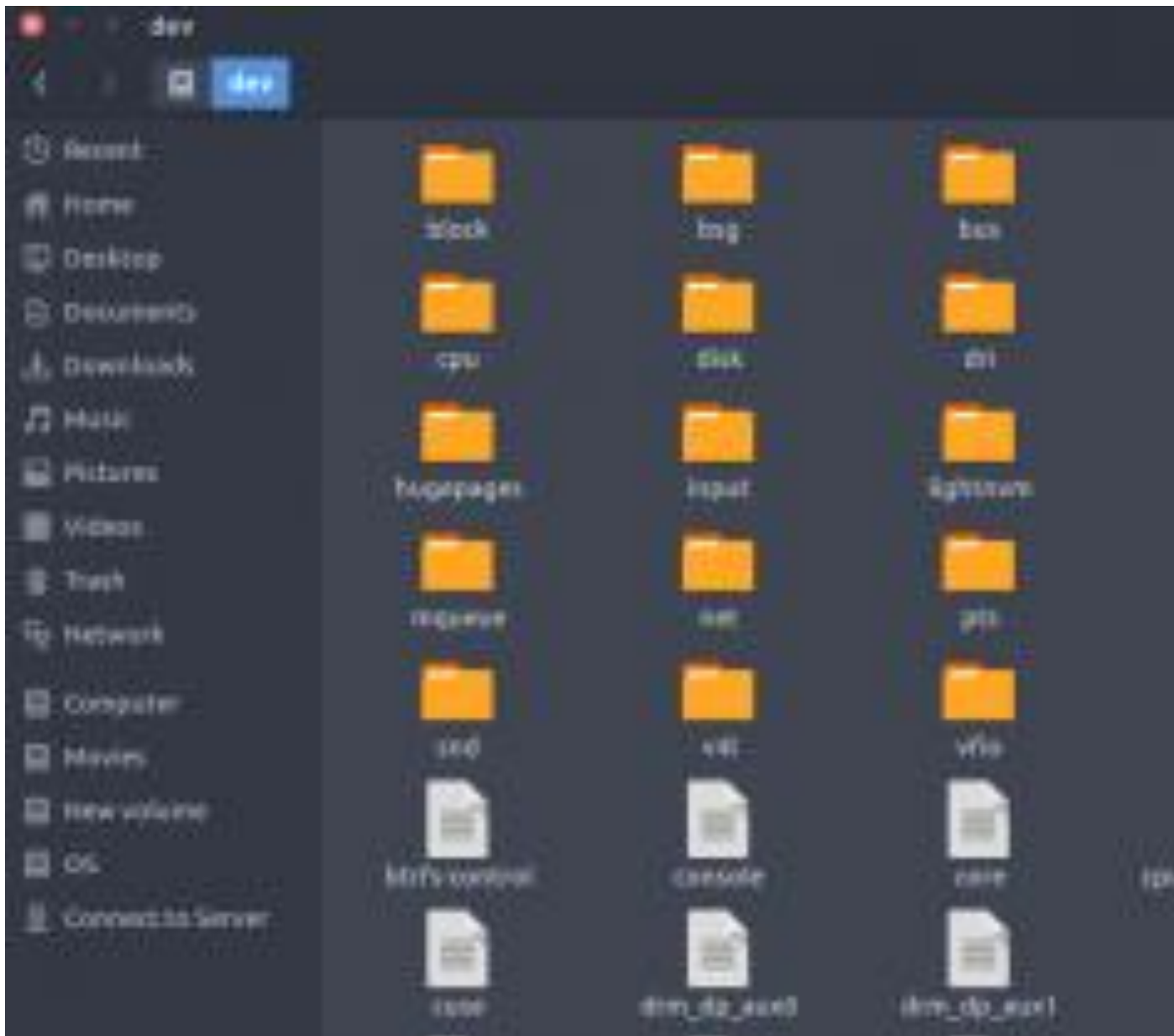
- [Contains binary executables](#)
- [Common linux commands you need to use in single-user modes are located under this directory.](#)
- [Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp](#)





**4. /dev**: Essential device files, e.g., /dev/null.

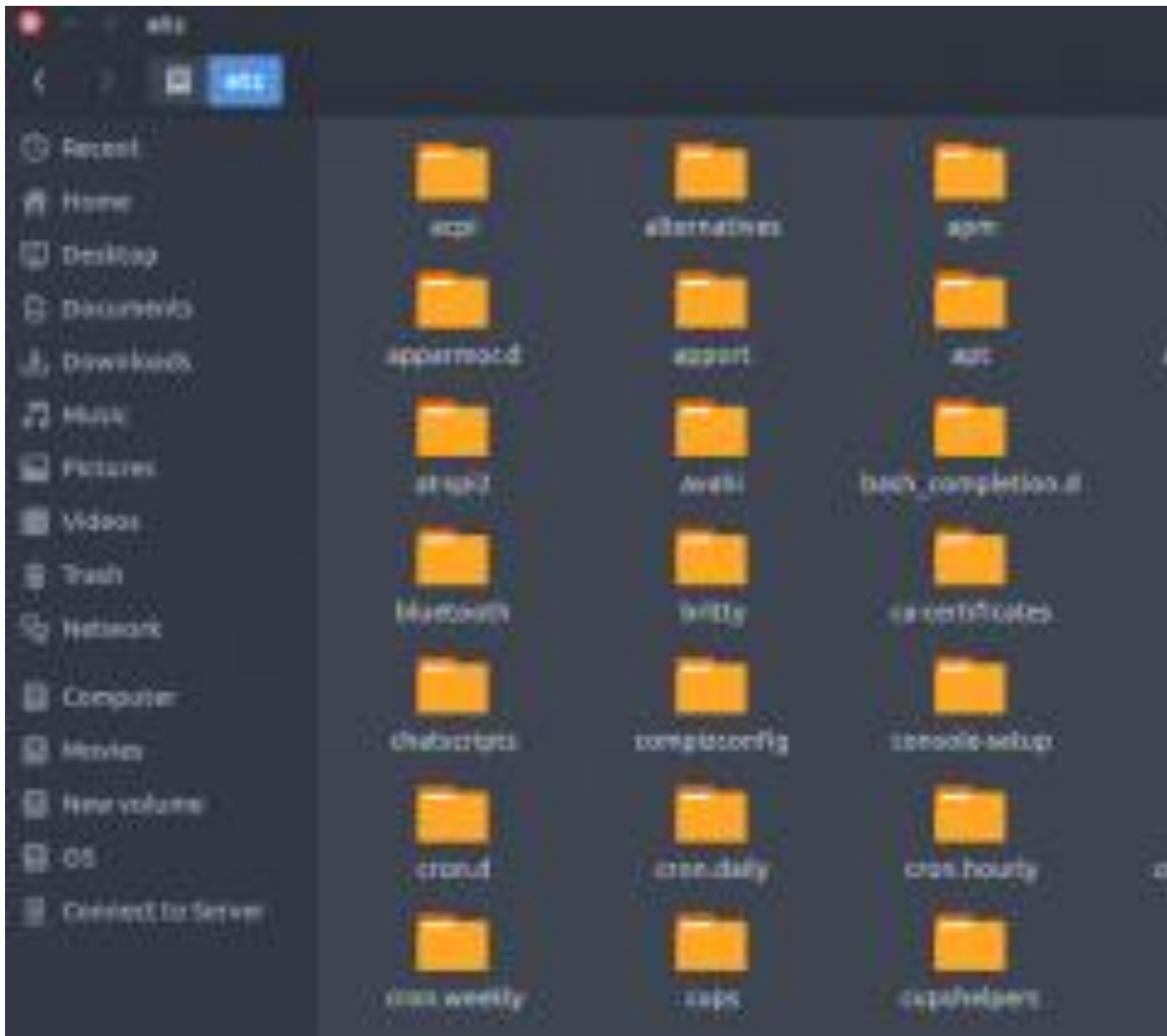
- These include terminal devices, usb, or any device attached to the system.
- Example: /dev/tty1, /dev/usbmon0



**5. /etc :** Host-specific system-wide configuration files.

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- Example: /etc/resolv.conf, /etc/logrotate.conf.

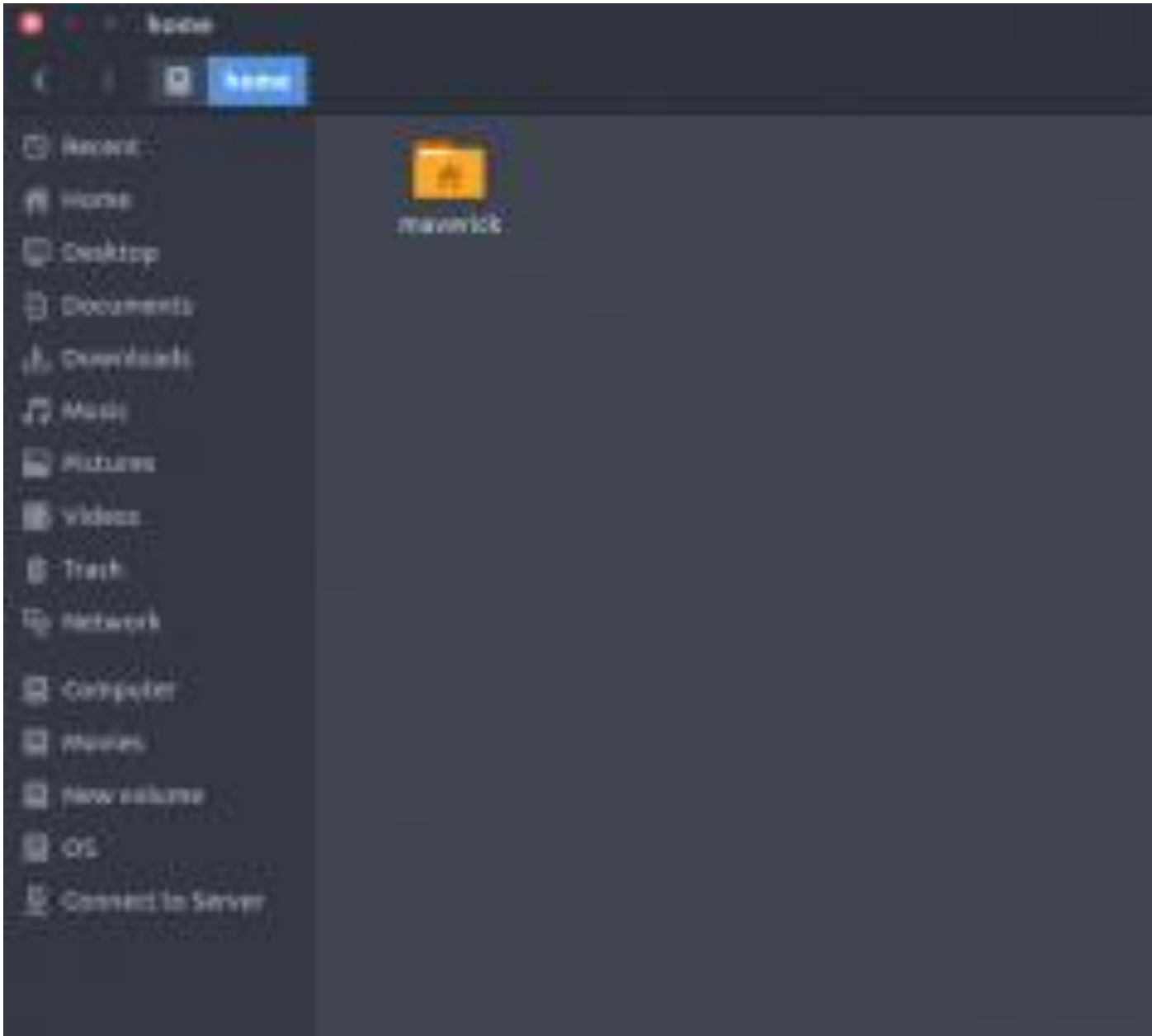




**6. /home :** Users' home directories, containing saved files, personal settings, etc.

- [Home directories for all users to store their personal files.](#)
- [example: /home/kishlay, /home/kv](#)





1. Bash Shell	2. Tcsh Shell	3. Ksh Shell
4. Zsh Shell	5. Fish Shell	

# 5 Most Frequently Used Shells for Linux

*5 Most Frequently Used Open Source Shells for Linux*

The shell is much more than just a command interpreter, it is also a programming language of its own with complete programming language constructs such as conditional execution, loops, variables, functions and many more.

That is why the Unix/GNU Linux shell is more powerful compared to the Windows shell.

[In this article, we shall take a look at some of the top most used open source shells on Unix/GNU Linux.](#)

## 1. Bash Shell

[Bash stands for Bourne Again Shell and it is the default shell on many Linux distributions today. It is also a sh-compatible shell and offers practical improvements over sh for programming and interactive use which includes:](#)

- [Command line editing](#)
- [Job Control](#)
- [Unlimited size command history](#)
- [Shell Functions and Aliases](#)
- [Unlimited size Indexed arrays](#)
- [Integer arithmetic in any base from two to sixty-four](#)

**NAME**

**bash** - GNU Bourne-Again Shell

**SYNOPSIS**

**bash** [options] [command\_string | file]

**COPYRIGHT**

Bash is Copyright (C) 1989-2013 by the Free Software Foundation

**DESCRIPTION**

**Bash** is an **sh**-compatible command language interpreter that executes commands read from the standard input or from a file. **Bash** also incorporates useful features from the Korn and C shells (**ksh** and **cs**).

**Bash** is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Std 1003.1). **Bash** can be configured to be POSIX-conformant by default.

**OPTIONS**

All of the single-character shell options documented in the Shell and Utilities portion of the **set** builtin command can be used as options when the

Manual page bash(1) line 1 (press h for help or q to quit)

*Bash Shell*

## 2. Tcsh/Csh Shell

Tcsh is enhanced C shell, it can be used as a interactive login shell and shell script command processor.

Tcsh has the following features:

- C like syntax
- Command-line editor
- Programmable word and filename completion
- Spelling correction
- Job control

**NAME**

`tcsh` - C shell with file name completion and command line editing

**SYNOPSIS**

```
tcsh [-bcdefFimnqstvVxX] [-Dname[=value]] [arg ...]
tcsh -l
```

**DESCRIPTION**

`tcsh` is an enhanced but completely compatible version of the UNIX C shell, `cs`(1). It is a command language interpreter used as an interactive login shell and a shell script command processor. It includes a command-line editor (see [The command-line editor](#)), programmable word completion (see [Completion and listing](#)), spelling correction (see [Spelling correction](#)), a history mechanism (see [History substitution](#)), job control (see [Jobs](#)) and a C-like syntax. The **NEW** section describes major enhancements of `tcsh` over `cs`(1). Throughout this manual, features of `tcsh` not found in most `cs`(1) implementations (specifically, the 4.4BSD `cs`) are labeled with ``(+)`', and features which are present in `cs`(1) but not usually documented are labeled with ``(u)`'.

Manual page `tcsh(1)` line 1 (press h for help or q to quit)

*Tcsh Shell*

### 3. Ksh Shell

[Ksh stands for Korn shell and was designed and developed by David G. Korn. It is a complete, powerful, high-level programming language and also an interactive command language just like many other Unix/GNU Linux shells.](#)

**NAME**

ksh, ksh93 - KornShell, a command and programming language

**SYNOPSIS**

```
ksh [ ±abcefhikmnoprstuvxBCDP ] [ -R file ] [ ±o option ] ...  
arg ... ]  
rksh [ ±abcefhikmnoprstuvxBCD ] [ -R file ] [ ±o option ] ...  
arg ... ]
```

**DESCRIPTION**

Ksh is a command and programming language that executes commands from a terminal or a file. Rksh is a restricted version of the interpreter ksh; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. Rpfksh is a profile shell version of the command interpreter ksh; it is used to execute commands with the attributes defined by the user's profiles (see pfexec(1)). See Invocation for the meaning of arguments to the shell.

**Definitions.**

A metacharacter is one of the following characters:

Manual page ksh(1) line 1 (press h for help or q to quit)

*Ksh Shell*

## 4. Zsh Shell

Zsh is designed to be interactive and it incorporates many features of other Unix/GNU Linux shells such as bash, tsh and ksh.

It is also a powerful scripting language just like the other shells available. Though it has some unique features that include:

- [Filename generation](#)
- [Startup files](#)
- [Login/Logout watching](#)
- [Closing comments](#)
- [Concept index](#)
- [Variable index](#)
- [Functions index](#)
- [Key index and many more that you can find out in man pages](#)

ZSH(1)

General Commands Manual

## NAME

zsh - the Z shell

## OVERVIEW

Because zsh contains many features, the zsh manual has been split into a number of sections:

<u>zsh</u>	Zsh overview (this section)
<u>zshroadmap</u>	Informal introduction to the manual
<u>zshmisc</u>	Anything not fitting into the other sections
<u>zshexpn</u>	Zsh command and parameter expansion
<u>zshparam</u>	Zsh parameters
<u>zshoptions</u>	Zsh options
<u>zshbuiltins</u>	Zsh built-in functions
<u>zshzle</u>	Zsh command line editing
<u>zshcompwid</u>	Zsh completion widgets
<u>zshcompsys</u>	Zsh completion system
<u>zshcompctl</u>	Zsh completion control
<u>zshmodules</u>	Zsh loadable modules
<u>zshcalsys</u>	Zsh built-in calendar functions
<u>zshtcpsys</u>	Zsh built-in TCP functions

Manual page zsh(1) line 1 (press h for help or q to quit)

*Zsh Shell*

## 5. Fish

[Fish](#) in full stands for “friendly interactive shell” and was authored in 2005. It was intended to be fully interactive and user friendly, just like the other shells, it has some pretty good features that include:

- [Man page completions](#)
- [Web based configuration](#)
- [Auto-suggestions](#)
- [Fully scriptable with clean scripts](#)
- [Support for term256 terminal technology](#)

You can read more about fish shell at [Fish – A Smart Interactive Shell for Linux](#)

```
fish(1)                                fish

NAME
    fish - fish - the friendly interactive shell

fish - the friendly interactive shell
Synopsis
    fish [-h] [-v] [-c command] [FILE [ARGUMENTS...]]

Description
    fish is a command-line shell written mainly with interactive use in
    mind. The full manual is available in HTML by using the help command
    from inside fish.

    The following options are available:

    · -c or --command=COMMANDS evaluate the specified commands instead of
      reading from the commandline

    · -d or --debug-level=DEBUG_LEVEL specify the verbosity level of the
      shell. A higher number means higher verbosity. The default level is 0.

    · -h or --help display help and exit

Manual page fish(1) line 1 (press h for help or q to quit)
```

## Unix Filesystem Organization

### ``Old'' (Original) file system

In the original Unix file system, Unix divided physical disks into logical disks called *partitions*. Each partition is a standalone file system. We will use the term ``file system'' when referring to a single partition.

Each disk device is given its own *major device number*, and each partition has an associated *minor device number* which the device driver uses to access the raw file system.

The major/minor device number combination serves as a handle into the device switch table. That is, the major number acts as an index, and the minor number is passed as an argument to the driver routines so that they can recognize the specific instance of a device.

Each filesystem contains:





An integral number of inodes fits in a single data block.

Information the inode does not contain:

- path (short or full) name of file

## Example

Look at /cs/bin/

```
< wpi /cs/bin 1 >ls -l
total 192
drwx-----  2 mvoorhis csadmin    4096 Jan 16  2001 archives/
-rws--x---  1 root      771      32768 Jan 18  1999 csquotamgr*
-rwx-----  1 csadmin  csadmin    162 Jan 12  1998 genQuota*
-rwx-----  1 csadmin  csadmin    46 Feb 16  1998 generic*
drwxrwx---  2 mvoorhis csadmin    4096 Oct 29 10:23 gredStuff/
-rwx-----  1 mvoorhis 1067      672 Jan 20  2000 list1*
-rwx-----  1 mvoorhis 1067      859 Jan 20  2000 list2*
-rwx-----  1 csadmin  646      140 Jan 10  2000 reclaim*
-rwxrwx---  1 csadmin  csadmin   1635 Sep 26  1995 stp_create_system.pl*
-rwxrwxr-x  1 csadmin  csadmin    725 Sep 26  1995
stp_default_system.pl*
-rw-rw-r--  1 csadmin  csadmin    114 Feb 10  1995 stp_setup
drwx----- 14 mvoorhis csadmin    4096 Oct 30 14:57 tDir/
-rwsr-xr-x  1 mvoorhis 1067   114688 Nov  8 10:10 turnin*
drwxr-xr-x  2 root      771      4096 May 26  1999 utility/
```

Internally, Unix stores directories in files. The file type (of the inode) is marked ``directory'', and the file contains pairs of name/inode numbers.

For example, when a user issues *open(``/etc/passwd'', ...)* the kernel performs the following operations:

1.

because the file name is a full path name, find the inode of the root directory (found in superblock) and search the corresponding file for the entry ``etc''

2.

when the entry ``etc'' is found, fetch its corresponding inode and check that it is of type directory

3.

scan the file associated with ``/etc'' looking for ``passwd''

4.

finally, fetch the inode associated with *passwd*'s directory entry, verify that it is a regular file, and start accessing the file.

Note: What would the system do when opening ```/dev/tty01"`?

Eventually, the system would find the inode corresponding to the device, and note that its file type was ```special"`. Thus, it would extract the major/minor device number pair from the length field of the inode, and use the device number as an index into the device switch table.

### Getwd()

How to get string of current directory? Have only the inode of the current directory.

```
get current inode
while (inode != root inode) {
    get inode of parent from ..
    search parent's directory file to match our inode number
```

Where should a file's data blocks be physically located?

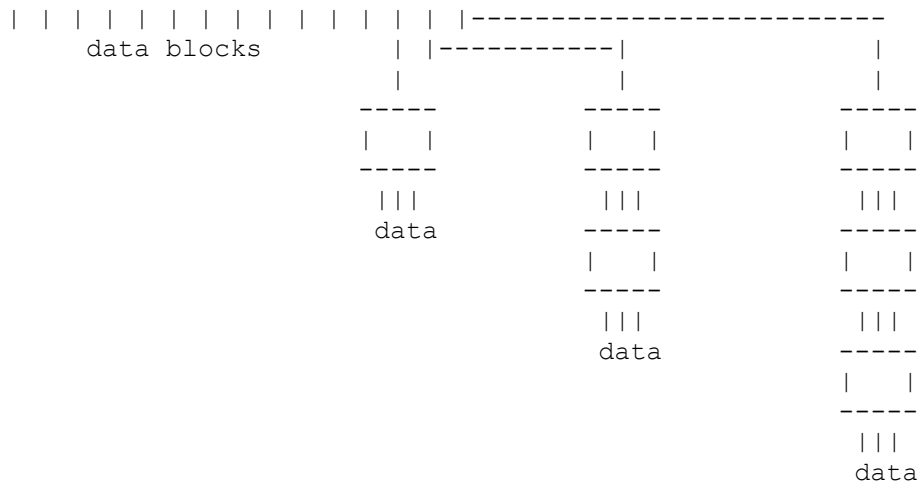
- to improve performance, we might want to place a file's data blocks in contiguous sectors on disk. However, this leads to inefficiencies in allocating space, or forces the user to specify the size of the file at creation time.

The Unix file system allocates *data blocks* (blocks that contain a file's contents) one at a time from a pool of free blocks. Unix uses 4K blocks. Moreover, a file's blocks are scattered randomly within the physical disk.

Inodes include pointers to the data blocks. Each inode contains 15 pointers:

- the first 12 pointers point directly to data blocks
- the 13th pointer points to an indirect block, a block containing pointers to data blocks
- the 14th pointer points to a doubly-indirect block, a block containing 128 addresses of singly indirect blocks
- the 15th pointer points to a triply indirect block (which contains pointers to doubly indirect blocks, etc.)





with 4K blocks:  
 direct  $12 \times 4K = 48K$   
 indirect  $1024 \times 4K = 4MB$   
 double indirect  $1024 \times 1024 \times 4K = 4GB$   
 triple indirect  $1024 \times 1024 \times 1024 \times 4K = 4TB$

### Advantages:

- data in small files can be accessed directly from the inode That is, one read operation fetches the inode, and another read fetches the first data block.
- larger files can be accessed efficiently, because an indirect block points to many data blocks
- disk can be filled completely, with little wasted space (ignoring partially-filled blocks)

### Disadvantages:

- because inode information is kept separately from data, access of data often requires a long seek when file is initially accessed
- inodes of files in a common directory not kept together, leading to low performance when searching directories
- original file system only used 512-byte blocks, an inefficient transfer size
- data blocks of a file are not stored together, leading to poor performance when accessing files sequentially.
- free list quickly becomes scrambled increasing overhead of finding free blocks (seek for each new block)
- original file system used as little as 2% of the available disk bandwidth

## The Berkeley Fast File System

The Berkeley Fast File System used the following principles to improve the performance (and reliability) of the file system:

- duplicate the super block, so that it can easily be recovered after a disk crash
- use a large block size to improve throughput
- add the block size to the superblock, so that different file systems could be accessed using different block sizes
- store related data blocks within *cylinder groups*, one or more consecutive cylinders on the disk. Blocks within in a cylinder group can be accessed with only a small seek (if any)
- because large blocks leads to fragmentation, small files (or the remaining bytes of a file) should be stored in *fragments*, where an integral number of fragments (e.g., 4 or 8) fits in a single block. Use 4K/512 or 8K/1K block/fragment size combinations. Inode stores 8-bit mask for fragment use in last data block of the file.

A data structure within each cylinder group contains status information about the blocks stored within that group:

1.

a bit map of blocks and fragments indicates which blocks and fragments are free

2.

a list of inodes within the cylinder group (why?)

3.

duplicate copy of the superblock (stored on a different platter for each cylinder group. why?)

When allocating space, the fast file system uses a *global policy* to determine where to place new directories and files. For example:

- place inodes of files in the same directory in the same cylinder group (makes programs like *ls* faster)
- place new directories in a cylinder group that has a higher than average number of free inodes and the smallest number of directories already in it
- try to place all data blocks for a file in the same cylinder group
- move to a new cylinder group when a file exceeds 48kb, and every megabyte thereafter.

The fast file system also uses a *local policy* to allocate blocks at the lower levels. For instance:

- when adding a data block to an existing file, pick the next block to be rotationally closest
- try to allocate blocks out of the same cylinder, before looking at other blocks in the same cylinder group

The new file system increased the throughput to as much as 30% of the raw bandwidth.

## Linux ext2fs

The *2nd extended file system*. Same standard file system as Unix.

Similar to the Berkeley fast file system, but does not use fragments. Rather it uses smaller block sizes (1K, but can be 2K or 4K).

Tries to cluster disk blocks so that a single I/O request can read multiple blocks.

Modern disk technologies pack sectors onto disks at different densities--Linux uses variable size block groups (like cylinder groups in BSD FFS).

Allocation:

- Tries to allocate data blocks in same group as inode.
- Tries to allocate nondirectory inodes in same group as parent directory
- Tries to allocate directory inodes in different group than parent directory

Also has a *proc* file system to allow access to process information through the file system interface.

Also supports other file systems such as FAT and NTFS.

## File Mounting

When the system initially boots, the only file system Unix knows about is the root partition from which the system was booted. A special system call:

*mount(special, path\_name, options)*

mounts the filesystem given by *special* at the point *path\_name* in the root filesystem, thus allowing multiple file systems to be merged into a single global tree.

Internally, the kernel maintains a *mount table* that keeps information about the mounted file systems. Each entry in the table contains:

- the device number of partition that has been mounted
- a pointer to the buffer containing the super block for the file system
- a pointer to the root inode of the file system
- a pointer to the inode of the directory in which the file system is mounted (e.g., a pointer to the parent directory)

As the kernel is translating a path name, it consults the mount table as needed.

## In Memory Data Structures

The kernel maintains a system-wide *file table* that describes open files. Each entry in the file table contains:

- the read/write mark that indicates from where the next data block is to be read
- a pointer to an entry in the *active inode table*, so that the access times can be modified efficiently

Finally, each process maintains a *user file table* that describes the files opened by the process. Entries in the user file table point to the system-wide file table.

Thus, a process can have its own private read/write mark (the default when a file is initially opened), or it can share a read/write mark (as is the case when a new process is created via *fork*).

## Caching

Unix relies heavily on caching to improve performance. It keeps in memory:

- recently accessed disk blocks
- the inode of each open file in the system
- a cache of recent name-to-inode mappings
- a directory offset cache. If a process requests a file name in the same directory as its previous request, the search through the directory is started where the previous search ended. Improves the performance of applications that read all files in a directory (like *ls*).
- The `grep` utilities are a family of [Unix](#) tools, including `grep`, `egrep`, and `fgrep`, that perform repetitive searching tasks. The tools in the `grep` family are very similar, and all are used for searching the contents of files for information that matches particular criteria. For most purposes, you'll want to use `fgrep`, since it's generally the fastest.
- The general syntax of the `grep` commands is:

- `grep [-options] pattern [filename]`

- You can use `fgrep` to find all the lines of a file that contain a particular word. For example, to list all the lines of a file named `myfile` in the current directory that contain the word "dog", enter at the Unix prompt:

```
• fgrep dog myfile
```

- This will also return lines where "dog" is embedded in larger words, such as "dogma" or "dogged". You can use the `-w` option with the `grep` command to return only lines where "dog" is included as a separate word:

```
• grep -w dog myfile
```

- To search for several words separated by spaces, enclose the whole search string in quotes, for example:

```
• fgrep "dog named Checkers" myfile
```

- The `fgrep` command is case sensitive; specifying "dog" will not match "Dog" or "DOG". You can use the `-i` option with the `grep` command to match both upper- and lowercase letters:

```
• grep -i dog myfile
```

- To list the lines of `myfile` that do not contain "dog", use the `-v` option:

```
• fgrep -v dog myfile
```

- If you want to search for lines that contain any of several different words, you can create a second file (named `secondfile` in the following example) that contains those words, and then use the `-f` option:

```
• fgrep -f secondfile myfile
```

- You can also use wildcards to instruct `fgrep` to search any files that match a particular pattern. For example, if you wanted to find lines containing "dog" in any of the files in your directory with names beginning with "my", you could enter:

```
• fgrep dog my*
```

- This command would search files with names such as `myfile`, `my.hw1`, and `mystuff` in the current directory. Each line returned will be prefaced with the name of the file where the match was found.

- By using pipes and/or redirection, you can use the output from any of these commands with other Unix tools, such as `more`, `sort`, and `cut`. For example, to print the fifth word of every line of `myfile` containing "dog", sort the words alphabetically, and then filter the output through the `more` command for easy reading, you would enter at the Unix prompt:

```
• fgrep dog myfile | cut -f5 -d" " | sort | more
```

- If you want to save the output in a file in the current directory named `newfile`, enter:

- `fgrep dog myfile | cut -f5 -d" " | sort > newfile`

## find command in Linux with examples

The **find** command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

### Syntax :

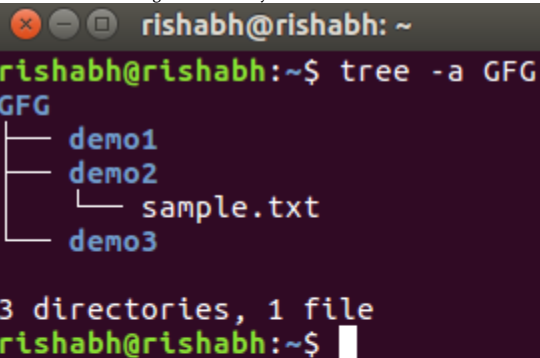
```
$ find [where to start searching from]
[expression determines what to find] [-options] [what to find]
```

### Options :

- **-exec CMD**: The file being searched which meets the above criteria and returns 0 for as its exit status for successful command execution.
- **-ok CMD**: It works same as -exec except the user is prompted first.
- **-inum N**: Search for files with inode number 'N'.
- **-links N**: Search for files with 'N' links.
- **-name demo**: Search for files that are specified by 'demo'.
- **-newer file**: Search for files that were modified/created after 'file'.
- **-perm octal**: Search for the file if permission is 'octal'.
- **-print**: Display the path name of the files found by using the rest of the criteria.
- **-empty**: Search for empty files and directories.
- **-size +N/-N**: Search for files of 'N' blocks; 'N' followed by 'c' can be used to measure size in characters; '+N' means size > 'N' blocks and '-N' means size < 'N' blocks.
- **-user name**: Search for files owned by user name or ID 'name'.
- **\(expr \)**: True if 'expr' is true; used for grouping criteria combined with OR or AND.
- **! expr**: True if 'expr' is false.

### Examples

Consider the following tree hierarchy :



```
rishabh@rishabh: ~
rishabh@rishabh:~$ tree -a GFG
GFG
├── demo1
├── demo2
│   └── sample.txt
└── demo3

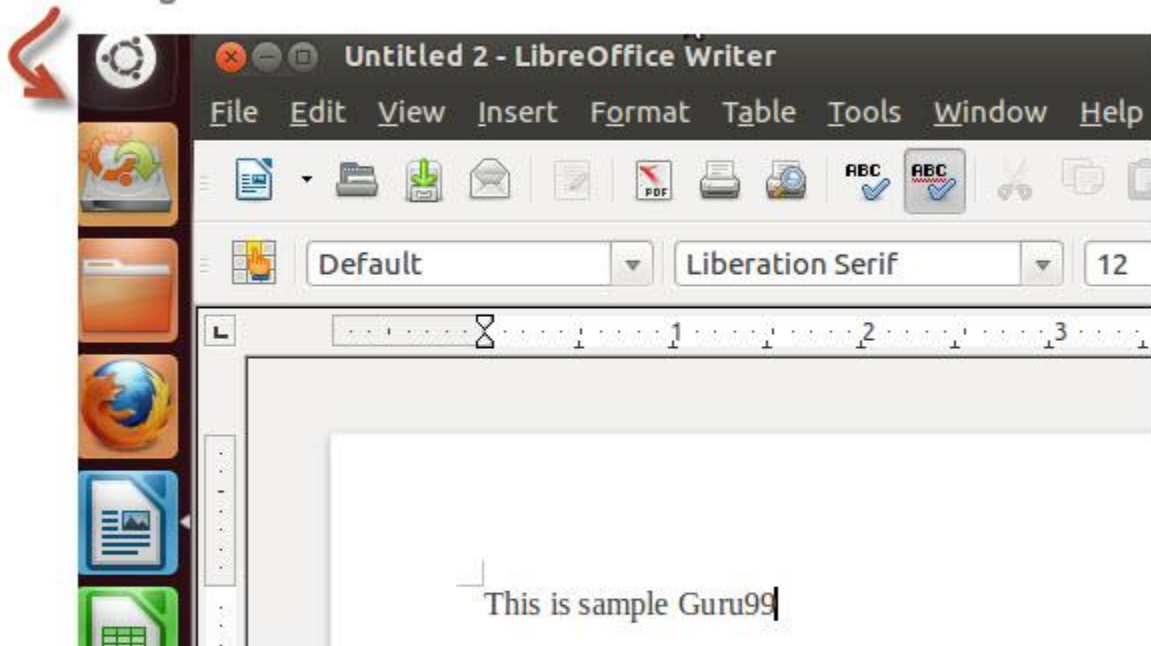
3 directories, 1 file
rishabh@rishabh:~$
```

What is a Process?



An instance of a program is called a Process. In simple terms, any command that you give to your Linux machine starts a new process.

When you launch Office to write some article



Corresponding process is created



Having multiple processes for the same program is possible.

Types of Processes:

- Foreground Processes: They run on the screen and need input from the user. For example Office Programs
- Background Processes: They run in the background and usually do not need user input. For example Antivirus.

Please be patient. The Video will load in some time. If you still face issue viewing video click [here](#)

## Running a Foreground Process

To start a foreground process, you can either run it from the dashboard, or you can run it from the terminal.

When using the Terminal, you will have to wait, until the foreground process runs.



OR

```
home@VirtualBox:~$ banshee
```

## Running a Background process

If you start a foreground program/process from the terminal, then you cannot work on the terminal, till the program is up and running.

Particular, data-intensive tasks take lots of processing power and may even take hours to complete. You do not want your terminal to be held up for such a long time.

To avoid such a situation, you can run the program and send it to the background so that terminal remains available to you. Let's learn how to do this -

*Start the program and press ctrl+z*

```
guru99@VirtualBox:~$ banshee
[Info 16:08:36.688] Running Banshee 2.2.1: [Ubuntu 11.
11-12-19 14:51:26 UTC]
^Z
[1]+  Stopped                  banshee
```

*Type 'bg' to send the process to the background*

```
guru99@VirtualBox:~$ bg
```

## Fg

You can use the command "fg" to continue a program which was stopped and bring it to the foreground.

The simple syntax for this utility is:

```
fg jobname
```

Example

1. Launch 'banshee' music player
2. Stop it with the 'ctrl +z' command
3. Continue it with the 'fg' utility.

```

home@VirtualBox:~$ banshee
^Z
[1]+  Stopped                  banshee
home@VirtualBox:~$ fg banshee
banshee
[Info 00:36:19.400] Running Banshee 2.2.0: [Ubuntu oneiric
(linux-gnu, i686) @ 2011-09-23 04:51:00 UTC]

```

Let's look at other important commands to manage processes -

## Top

This utility tells the user about all the running processes on the Linux machine.

```

home@VirtualBox:~$ top
top - 23:57:43 up 2:54, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 189 total, 2 running, 187 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 3.0%sy, 0.0%ni, 96.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1026080k total, 924508k used, 101572k free, 37000k buffers
Swap: 1046524k total, 21472k used, 1025052k free, 367996k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1525 home      20   0 1775m 100m  28m  S   1.7  10.0   5:05.34 Photoshop.exe
  961 root      20   0 75972  51m 7952  R   1.0   5.1   2:23.42 Xorg
 1507 home      20   0  7644 4652  696  S   1.0   0.5   2:42.66 wineserver
 1564 home      20   0 75144  29m 9840  S   0.3   3.0   0:25.96 ubuntuone-syncd
 2999 home      20   0  127m  13m  10m  S   0.3   1.4   0:01.36 gnome-terminal
 3077 home      20   0  2820  1188 864  R   0.3   0.1   0:00.76 top
    1 root      20   0  3200  1704 1260  S   0.0   0.2   0:00.98 init
    2 root      20   0    0    0    0  S   0.0   0.0   0:00.00 kthreadd
    3 root      20   0    0    0    0  S   0.0   0.0   0:00.95 ksoftirqd/0

```

Press 'q' on the keyboard to move out of the process display.

The terminology follows:

Field	Description	Example 1	Example 2
PID	The process ID of each task	1525	961
User	The username of task owner	Home	Root
PR	Priority Can be 20(highest) or -20(lowest)	20	20
NI	The nice value of a task	0	0
VIRT	Virtual memory used (kb)	1775	75972
RES	Physical memory used (kb)	100	51
SHR	Shared memory used (kb)	28	7952

Status

S            There are five types:            S            R

'D' = uninterruptible sleep

Field	Description	Example 1	Example 2
	'R' = running		
	'S' = sleeping		
	'T' = traced or stopped		
	'Z' = zombie		
%CPU	% of CPU time	1.7	1.0
%MEM	Physical memory used	10	5.1
TIME+	Total CPU time	5:05.34	2:23.42
Command	Command name	Photoshop.exe	Xorg

## PS

This command stands for 'Process Status'. It is similar to the "Task Manager" that pop-ups in a Windows Machine when we use Cntrl+Alt+Del. This command is similar to 'top' command but the information displayed is different.

To check all the processes running under a user, use the command -

```
ps ux
```

```
home@VirtualBox:~$ ps ux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
home     1114  0.0  0.8  46548  8512 ?        Ssl   Sep03   0:00 gnome-sess
home     1151  0.0  0.0   3856   140 ?        Ss    Sep03   0:00 /usr/bin/s
home     1154  0.0  0.0   3748   484 ?        S     Sep03   0:00 /usr/bin/d
home     1155  0.1  0.2   6656  3036 ?        Ss    Sep03   0:18 //bin/dbus
home     1157  0.0  0.2   9148  2368 ?        S     Sep03   0:00 /usr/lib/g
home     1162  0.0  0.2  31588  2296 ?        Ssl   Sep03   0:00 /usr/lib/g
home     1174  0.0  1.4 132472 14884 ?        Sl    Sep03   0:03 /usr/lib/g
```

You can also check the process status of a single process, use the syntax -

```
ps PID
```

```
guru99@VirtualBox:~$ ps 1268
  PID TTY      STAT   TIME COMMAND
 1268 ?        S<l    0:02 /usr/bin/pulseaudio --start --log-target=syslog
```

## Kill

This command **terminates running processes** on a Linux machine.

To use these utilities you need to know the PID (process id) of the process you want to kill

Syntax -

```
kill PID
```

To find the PID of a process simply type

```
pidof Process name
```

Let us try it with an example.

```
home@VirtualBox:~$ pidof Photoshop.exe
1525
home@VirtualBox:~$ kill 1525
```

## NICE

Linux can run a lot of processes at a time, which can slow down the speed of some high priority processes and result in poor performance.

To avoid this, you can tell your machine to prioritize processes as per your requirements.

This priority is called Niceness in Linux, and it has a value between -20 to 19. The lower the Niceness index, the higher would be a priority given to that task.

The default value of all the processes is 0.

To start a process with a niceness value other than the default value use the following syntax

```
nice -n 'Nice value' process name
```

```
home@VirtualBox:~$ nice -n 19 banshee
```

If there is some process already running on the system, then you can 'Renice' its value using syntax.

```
renice 'nice value' -p 'PID'
```

To change Niceness, you can use the 'top' command to determine the PID (process id) and its Nice value. Later use the renice command to change the value.

Let us understand this by an example.

*Checking the niceness value of the process 'banshee'*

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3293	home	20	0	277m	64m	35m	S	96.4	6.4	9:56.72	banshee

*Renicing the value to -20*

```
home@VirtualBox:~$ sudo renice -20 -p 3293
[sudo] password for home:
3293 (process ID) old priority 0, new priority -20
```

*The value changed to -20*

3293	home	0	-20	277m	64m	35m	S	95.2	6.4	3:32.95	banshee
------	------	---	-----	------	-----	-----	---	------	-----	---------	---------

## DF

This utility reports the free disk space(Hard Disk) on all the file systems.

```

guru99@guru99-VirtualBox:~$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        7837756 2921376   4523216  40% /
udev             246488      4     246484   1% /dev
tmpfs            101512      752    100760   1% /run
none              5120        0         5120   0% /run/lock
none             253776      76     253700   1% /run/shm

```

If you want the above information in a readable format, then use the command

```
'df -h'
```

```

guru99@guru99-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        7.5G  2.8G  4.4G  40% /
udev             241M  4.0K  241M   1% /dev
tmpfs            100M  752K   99M   1% /run
none             5.0M    0   5.0M   0% /run/lock
none            248M   76K  248M   1% /run/shm

```

## Free

This command shows the free and used memory (RAM) on the Linux system.

```

home@VirtualBox:~$ free
              total        used         free       shared    buffers     cached
Mem:           1026080      803604      222476           0        36312     343376
-/+ buffers/cache:      423916        602164
Swap:          1046524       35832     1010692

```

You can use the arguments

free -m to display output in MB

free -g to display output in GB

## S

In Unix, there are three basic types of files –

- **Ordinary Files** – An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** – Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** – Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

## Listing Files

To list the files and directories stored in the current directory, use the following command –

```
$ls
```

Here is the sample output of the above command –

```
$ls
bin      hosts  lib    res.03
ch07    hw1    pub    test_results
ch07.bak hw2    res.01 users
docs     hw3    res.02 work
```

The command **ls** supports the **-l** option which would help you to get more information about the listed files –

```
$ls -l
total 1962188
drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml
-rw-rw-r-- 1 amrood amrood 5341 Dec 25 08:38 uml.jpg
drwxr-xr-x 2 amrood amrood 4096 Feb 15 2006 univ
drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia
-rw-r--r-- 1 root root 276480 Dec 9 2007 urlspedia.tar
drwxr-xr-x 8 root root 4096 Nov 25 2007 usr
drwxr-xr-x 2 200 300 4096 Nov 25 2007 webthumb-1.01
-rw-rw-r-- 1 root root 3192 Nov 25 2007 webthumb.php
-rw-rw-r-- 1 amrood amrood 20480 Nov 25 2007 webthumb.tar
-rw-rw-r-- 1 amrood amrood 5654 Aug 9 2007 yourfile.mid
-rw-rw-r-- 1 amrood amrood 166255 Aug 9 2007 yourfile.swf
drwxr-xr-x 11 amrood amrood 4096 May 29 2007 zlib-1.2.3
$
```

Here is the information about all the listed columns –

- **First Column** – Represents the file type and the permission given on the file. Below is the description of all type of files.
- **Second Column** – Represents the number of memory blocks taken by the file or directory.
- **Third Column** – Represents the owner of the file. This is the Unix user who created this file.
- **Fourth Column** – Represents the group of the owner. Every Unix user will have an associated group.
- **Fifth Column** – Represents the file size in bytes.
- **Sixth Column** – Represents the date and the time when this file was created or modified for the last time.
- **Seventh Column** – Represents the file or the directory name.

In the **ls -l** listing example, every file line begins with a **d**, **-**, or **l**. These characters indicate the type of the file that's listed.

Sr.No.	Prefix & Description
1	<b>-</b> Regular file, such as an ASCII text file, binary executable, or hard link.
2	<b>b</b> Block special file. Block input/output device file such as a physical hard drive.
3	<b>c</b> Character special file. Raw input/output device file such as a physical hard drive.
4	<b>d</b> Directory file that contains a listing of other files and directories.
5	<b>l</b> Symbolic link file. Links on any regular file.

6	<b>p</b> Named pipe. A mechanism for interprocess communications.
7	<b>s</b> Socket used for interprocess communication.

## Metacharacters

Metacharacters have a special meaning in Unix. For example, **\*** and **?** are metacharacters. We use **\*** to match 0 or more characters, a question mark (**?**) matches with a single character.

For Example –

```
$ls ch*.doc
```

Displays all the files, the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc  ch010.doc  ch02.doc  ch03-2.doc
ch04-1.doc  ch040.doc  ch05.doc  ch06-2.doc
ch01-2.doc  ch02-1.doc  c
```

Here, **\*** works as meta character which matches with any character. If you want to display all the files ending with just **.doc**, then you can use the following command –

```
$ls *.doc
```

## Hidden Files

An invisible file is one, the first character of which is the dot or the period character (**.**). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files –

- **.profile** – The Bourne shell ( **sh** ) initialization script
- **.kshrc** – The Korn shell ( **ksh** ) initialization script
- **.cshrc** – The C shell ( **csh** ) initialization script
- **.rhosts** – The remote shell configuration file

To list the invisible files, specify the **-a** option to **ls** –

```
$ ls -a
.      .profile  docs    lib     test_results
..     .rhosts   hosts   pub     users
.emacs bin       hw1     res.01  work
.exrc  ch07     hw2     res.02
.kshrc ch07.bak hw3     res.03
$
```

- **Single dot (.)** – This represents the current directory.
- **Double dot (..)** – This represents the parent directory.

## Creating Files



You can use the **vi** editor to create ordinary files on any Unix system. You simply need to give the following command –

```
$ vi filename
```

The above command will open a file with the given filename. Now, press the key **i** to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file as in the following program –

```
This is unix file....I created it for the first time....  
I'm going to save this content in this file.
```

Once you are done with the program, follow these steps –

- Press the key **esc** to come out of the edit mode.
- Press two keys **Shift + ZZ** together to come out of the file completely.

You will now have a file created with **filename** in the current directory.

```
$ vi filename  
$
```

## Editing Files

You can edit an existing file using the **vi** editor. We will discuss in short how to open an existing file –

```
$ vi filename
```

Once the file is opened, you can come in the edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file –

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move upside in the file.
- **j** key to move downside in the file.

So using the above keys, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the **i** key to come in the edit mode. Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

## Display Content of a File

You can use the **cat** command to see the content of a file. Following is a simple example to see the content of the above created file –

```
$ cat filename  
This is unix file....I created it for the first time....  
I'm going to save this content in this file.  
$
```

You can display the line numbers by using the **-b** option along with the **cat** command as follows –

```
$ cat -b filename  
1 This is unix file....I created it for the first time....
```

```
2 I'm going to save this content in this file.  
$
```

## Counting Words in a File

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above –

```
$ wc filename  
2 19 103 filename  
$
```

Here is the detail of all the four columns –

- **First Column** – Represents the total number of lines in the file.
- **Second Column** – Represents the total number of words in the file.
- **Third Column** – Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column** – Represents the file name.

You can give multiple files and get information about those files at a time. Following is simple syntax –

```
$ wc filename1 filename2 filename3
```

## Copying Files

To make a copy of a file use the **cp** command. The basic syntax of the command is –

```
$ cp source_file destination_file
```

Following is the example to create a copy of the existing file **filename**.

```
$ cp filename copyfile  
$
```

You will now find one more file **copyfile** in your current directory. This file will exactly be the same as the original file **filename**.

## Renaming Files

To change the name of a file, use the **mv** command. Following is the basic syntax –

```
$ mv old_file new_file
```

The following program will rename the existing file **filename** to **newfile**.

```
$ mv filename newfile  
$
```

The **mv** command will move the existing file completely into the new file. In this case, you will find only **newfile** in your current directory.

## Deleting Files

To delete an existing file, use the **rm** command. Following is the basic syntax –

```
$ rm filename
```

**Caution** – A file may contain useful information. It is always recommended to be careful while using this **Delete** command. It is better to use the **-i** option along with **rm** command.

Following is the example which shows how to completely remove the existing file **filename**.

```
$ rm filename
$
```

You can remove multiple files at a time with the command given below –

```
$ rm filename1 filename2 filename3
$
```

## Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up –

- **stdin** – This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
- **stdout** – This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
- **stderr** – This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.
- In this chapter, we will discuss in detail about directory management in Unix.
- A directory is a file the solo job of which is to store the file names and the related information. All the files, whether ordinary, special, or directory, are contained in directories.
- Unix uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree. The tree has a single root node, the slash character (/), and all other directories are contained below it.

## • Home Directory

- The directory in which you find yourself when you first login is called your home directory.
- You will be doing much of your work in your home directory and subdirectories that you'll be creating to organize your files.
- You can go in your home directory anytime using the following command –

```
• $cd ~
• $
```

- Here **~** indicates the home directory. Suppose you have to go in any other user's home directory, use the following command –

```
• $cd ~username
• $
```

- To go in your last directory, you can use the following command –

```
• $cd -  
• $
```

## • Absolute/Relative Pathnames

- Directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname.
- Elements of a pathname are separated by a /. A pathname is absolute, if it is described in relation to root, thus absolute pathnames always begin with a /.
- Following are some examples of absolute filenames.

```
• /etc/passwd  
• /users/sjones/chem/notes  
• /dev/rdsk/0s3
```

- A pathname can also be relative to your current working directory. Relative pathnames never begin with /. Relative to user amrood's home directory, some pathnames might look like this –

```
• chem/notes  
• personal/res
```

- To determine where you are within the filesystem hierarchy at any time, enter the command **pwd** to print the current working directory –

```
• $pwd  
• /user0/home/amrood  
•  
• $
```

## • Listing Directories

- To list the files in a directory, you can use the following syntax –

```
• $ls dirname
```

- Following is the example to list all the files contained in **/usr/local** directory –

```
• $ls /usr/local  
•  
• X11      bin      gimp     jikes    sbin  
• ace      doc      include  lib       share  
• atalk    etc      info     man       ami
```

## • Creating Directories

- We will now understand how to create directories. Directories are created by the following command –

```
• $mkdir dirname
```

- Here, `directory` is the absolute or relative pathname of the directory you want to create. For example, the command –

```
• $mkdir mydir
• $
```

- Creates the directory **mydir** in the current directory. Here is another example –

```
• $mkdir /tmp/test-dir
• $
```

- This command creates the directory **test-dir** in the **/tmp** directory. The **mkdir** command produces no output if it successfully creates the requested directory.
- If you give more than one directory on the command line, **mkdir** creates each of the directories. For example, –

```
• $mkdir docs pub
• $
```

- Creates the directories `docs` and `pub` under the current directory.

## • Creating Parent Directories

- We will now understand how to create parent directories. Sometimes when you want to create a directory, its parent directory or directories might not exist. In this case, **mkdir** issues an error message as follows –

```
• $mkdir /tmp/amrood/test
• mkdir: Failed to make directory "/tmp/amrood/test";
• No such file or directory
• $
```

- In such cases, you can specify the **-p** option to the **mkdir** command. It creates all the necessary directories for you. For example –

```
• $mkdir -p /tmp/amrood/test
• $
```

- The above command creates all the required parent directories.

## • Removing Directories

- Directories can be deleted using the **rmdir** command as follows –

```
• $rmdir dirname
• $
```

- **Note** – To remove a directory, make sure it is empty which means there should not be any file or sub-directory inside this directory.
- You can remove multiple directories at a time as follows –

```
• $rmdir dirname1 dirname2 dirname3
• $
```

- The above command removes the directories `dirname1`, `dirname2`, and `dirname3`, if they are empty. The **rmdir** command produces no output if it is successful.

## • Changing Directories

- You can use the **cd** command to do more than just change to a home directory. You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as given below –

```
• $cd dirname
• $
```

- Here, **dirname** is the name of the directory that you want to change to. For example, the command –

```
• $cd /usr/local/bin
• $
```

- Changes to the directory **/usr/local/bin**. From this directory, you can **cd** to the directory **/usr/home/amrood** using the following relative path –

```
• $cd ../../home/amrood
• $
```

## • Renaming Directories

- The **mv (move)** command can also be used to rename a directory. The syntax is as follows –

```
• $mv olddir newdir
• $
```

- You can rename a directory **mydir** to **yourdir** as follows –

```
• $mv mydir yourdir
• $
```

## • The directories **.** (dot) and **..** (dot dot)

- The **filename .** (dot) represents the current working directory; and the **filename ..** (dot dot) represents the directory one level above the current working directory, often referred to as the parent directory.
- If we enter the command to show a listing of the current working directories/files and use the **-a option** to list all the files and the **-l option** to provide the long listing, we will receive the following result.

```
• $ls -la
• drwxrwxr-x  4  teacher  class   2048  Jul 16 17:56 .
• drwxr-xr-x 60    root          1536  Jul 13 14:18 ..
• -----  1  teacher  class   4210  May 1 08:27 .profile
• -rwxr-xr-x  1  teacher  class   1948  May 12 13:42 memo
```

• \$

In this chapter, we will discuss in detail about file permission and access modes in Unix. File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes –

- **Owner permissions** – The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** – The permissions for others indicate what action all other users can perform on the file.

## The Permission Indicators

While using **ls -l** command, it displays various information related to file permission as follows –

```
$ls -l /home/amrood
-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile
drwxr-xr-- 1 amrood users 1024 Nov 2 00:10 mydir
```

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –

- The first three characters (2-4) represent the permissions for the file's owner. For example, **-rwxr-xr--** represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rwxr-xr--** represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rwxr-xr--** represents that there is **read (r)** only permission.

## File Access Modes

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the **read**, **write**, and **execute** permissions, which have been described below –

### Read

Grants the capability to read, i.e., view the contents of the file.

### Write

Grants the capability to modify, or remove the content of the file.

### Execute

User with execute permissions can run a file as a program.

## Directory Access Modes

Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned –

## Read

Access to a directory means that the user can read the contents. The user can look at the **filenames** inside the directory.

## Write

Access means that the user can add or delete files from the directory.

## Execute

Executing a directory doesn't really make sense, so think of this as a traverse permission.

A user must have **execute** access to the **bin** directory in order to execute the **ls** or the **cd** command.

# Changing Permissions

To change the file or the directory permissions, you use the **chmod** (change mode) command. There are two ways to use chmod – the symbolic mode and the absolute mode.

## Using chmod in Symbolic Mode

The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.

Sr.No.	Chmod operator & Description
1	<b>+</b> Adds the designated permission(s) to a file or directory.
2	<b>-</b> Removes the designated permission(s) from a file or directory.
3	<b>=</b> Sets the designated permission(s).

Here's an example using **testfile**. Running **ls -l** on the testfile shows that the file's permissions are as follows –

```
$ls -l testfile
-rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
```

Then each example **chmod** command from the preceding table is run on the testfile, followed by **ls -l**, so you can see the permission changes –

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod u-x testfile
```



```
$ls -l testfile
-rw-rwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod g = rx testfile
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Here's how you can combine these commands on a single line –

```
$chmod o+wx,u-x,g = rx testfile
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

## Using chmod with Absolute Permissions

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.

Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Here's an example using the testfile. Running **ls -l** on the testfile shows that the file's permissions are as follows –

```
$ls -l testfile
-rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
```

Then each example **chmod** command from the preceding table is run on the testfile, followed by **ls -l**, so you can see the permission changes –

```
$ chmod 755 testfile
$ls -l testfile
-rwxr-xr-x 1 amrood users 1024 Nov 2 00:10 testfile
$chmod 743 testfile
$ls -l testfile
-rwxr---wx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod 043 testfile
$ls -l testfile
----r---wx 1 amrood users 1024 Nov 2 00:10 testfile
```

## Changing Owners and Groups

While creating an account on Unix, it assigns a **owner ID** and a **group ID** to each user. All the permissions mentioned above are also assigned based on the Owner and the Groups.

Two commands are available to change the owner and the group of files

–

- **chown** – The **chown** command stands for "**change owner**" and is used to change the owner of a file.
- **chgrp** – The **chgrp** command stands for "**change group**" and is used to change the group of a file.

## Changing Ownership

The **chown** command changes the ownership of a file. The basic syntax is as follows –

```
$ chown user filelist
```

The value of the user can be either the **name of a user** on the system or the **user id (uid)** of a user on the system.

The following example will help you understand the concept –

```
$ chown amrood testfile
$
```

Changes the owner of the given file to the user **amrood**.

**NOTE** – The super user, root, has the unrestricted capability to change the ownership of any file but normal users can change the ownership of only those files that they own.

## Changing Group Ownership

The **chgrp** command changes the group ownership of a file. The basic syntax is as follows –

```
$ chgrp group filelist
```

The value of group can be the **name of a group** on the system or **the group ID (GID)** of a group on the system.

Following example helps you understand the concept –

```
$ chgrp special testfile
$
```

Changes the group of the given file to **special** group.

## SUID and SGID File Permission

Often when a command is executed, it will have to be executed with special privileges in order to accomplish its task.

As an example, when you change your password with the **passwd** command, your new password is stored in the file **/etc/shadow**.

As a regular user, you do not have **read** or **write** access to this file for security reasons, but when you change your password, you need to have the write permission to this file. This means that the **passwd** program has to give you additional permissions so that you can write to the file **/etc/shadow**.

Additional permissions are given to programs via a mechanism known as the **Set User ID (SUID)** and **Set Group ID (SGID)** bits.

When you execute a program that has the SUID bit enabled, you inherit the permissions of that program's owner. Programs that do not have the SUID bit set are run with the permissions of the user who started the program.

This is the case with SGID as well. Normally, programs execute with your group permissions, but instead your group will be changed just for this program to the group owner of the program.

The SUID and SGID bits will appear as the letter "**s**" if the permission is available. The SUID "**s**" bit will be located in the permission bits where the owners' **execute** permission normally resides.

For example, the command –

```
$ ls -l /usr/bin/passwd
-r-sr-xr-x 1 root bin 19031 Feb 7 13:47 /usr/bin/passwd*
```

Shows that the SUID bit is set and that the command is owned by the root. A capital letter **S** in the execute position instead of a lowercase **s** indicates that the execute bit is not set.

If the sticky bit is enabled on the directory, files can only be removed if you are one of the following users –

- The owner of the sticky directory
- The owner of the file being removed
- The super user, root

To set the SUID and SGID bits for any directory try the following command –

```
$ chmod ug+s dirname
$ ls -l
drwsr-sr-x 2 root root 4096 Jun 19 06:45 dirname
```

In this chapter, we will discuss in detail about the Unix environment. An important Unix concept is the **environment**, which is defined by environment variables. Some are set by the system, others by you, yet others by the shell, or any program that loads another program.

A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

For example, first we set a variable TEST and then we access its value using the **echo** command –

```
$TEST="Unix Programming"
$echo $TEST
```

It produces the following result.

```
Unix Programming
```

Note that the environment variables are set without using the **\$** sign but while accessing them we use the **\$** sign as prefix. These variables retain their values until we come out of the shell.

When you log in to the system, the shell undergoes a phase called **initialization** to set up the environment. This is usually a two-step process that involves the shell reading the following files –

- `/etc/profile`
- `profile`

The process is as follows –

- The shell checks to see whether the file `/etc/profile` exists.
- If it exists, the shell reads it. Otherwise, this file is skipped. No error message is displayed.
- The shell checks to see whether the file `.profile` exists in your home directory. Your home directory is the directory that you start out in after you log in.
- If it exists, the shell reads it; otherwise, the shell skips it. No error message is displayed.

As soon as both of these files have been read, the shell displays a prompt –

```
$
```

This is the prompt where you can enter commands in order to have them executed.

**Note** – The shell initialization process detailed here applies to all **Bourne** type shells, but some additional files are used by **bash** and **ksh**.

## The `.profile` File

The file `/etc/profile` is maintained by the system administrator of your Unix machine and contains shell initialization information required by all users on a system.

The file `.profile` is under your control. You can add as much shell customization information as you want to this file. The minimum set of information that you need to configure includes –

- The type of terminal you are using.
- A list of directories in which to locate the commands.
- A list of variables affecting the look and feel of your terminal.

You can check your `.profile` available in your home directory. Open it using the vi editor and check all the variables set for your environment.

## Setting the Terminal Type

Usually, the type of terminal you are using is automatically configured by either the **login** or **getty** programs. Sometimes, the auto configuration process guesses your terminal incorrectly.

If your terminal is set incorrectly, the output of the commands might look strange, or you might not be able to interact with the shell properly.

To make sure that this is not the case, most users set their terminal to the lowest common denominator in the following way –

```
$TERM=vt100
$
```

## Setting the PATH

When you type any command on the command prompt, the shell has to locate the command before it can be executed.

The PATH variable specifies the locations in which the shell should look for commands. Usually the Path variable is set as follows –

```
$PATH=/bin:/usr/bin
$
```

Here, each of the individual entries separated by the colon character (**:**) are directories. If you request the shell to execute a command and it cannot find it in any of the directories given in the PATH variable, a message similar to the following appears –

```
$hello
hello: not found
$
```

There are variables like PS1 and PS2 which are discussed in the next section.

## PS1 and PS2 Variables

The characters that the shell displays as your command prompt are stored in the variable PS1. You can change this variable to be anything you want. As soon as you change it, it'll be used by the shell from that point on.

For example, if you issued the command –

```
$PS1='=>'
=>
=>
=>
```

Your prompt will become =>. To set the value of **PS1** so that it shows the working directory, issue the command –

```
=>PS1="\u@\h \w\"$"
[root@ip-72-167-112-17 /var/www/tutorialspoint/unix]$
[root@ip-72-167-112-17 /var/www/tutorialspoint/unix]$
```

The result of this command is that the prompt displays the user's username, the machine's name (hostname), and the working directory.

There are quite a few **escape sequences** that can be used as value arguments for PS1; try to limit yourself to the most critical so that the prompt does not overwhelm you with information.

Sr.No.	Escape Sequence & Description
1	<b>\t</b> Current time, expressed as HH:MM:SS
2	<b>\d</b> Current date, expressed as Weekday Month Date
3	<b>\n</b> Newline
4	<b>\s</b> Current shell environment
5	<b>\w</b> Working directory
6	<b>\W</b> Full path of the working directory
7	<b>\u</b> Current user's username
8	<b>\h</b> Hostname of the current machine
9	<b>\#</b> Command number of the current command. Increases when a new command is entered
10	<b>\\$</b> If the effective UID is 0 (that is, if you are logged in as root), end the prompt with the # character; otherwise, use the \$ sign

You can make the change yourself every time you log in, or you can have the change made automatically in PS1 by adding it to your **.profile** file.

When you issue a command that is incomplete, the shell will display a secondary prompt and wait for you to complete the command and hit **Enter** again.

The default secondary prompt is **>** (the greater than sign), but can be changed by re-defining the **PS2** shell variable –

Following is the example which uses the default secondary prompt –

```
$ echo "this is a
> test"
```

```
this is a
test
$
```

The example given below re-defines PS2 with a customized prompt –

```
$ PS2="secondary prompt->"
$ echo "this is a
secondary prompt->test"
this is a
test
$
```

## Environment Variables

Following is the partial list of important environment variables. These variables are set and accessed as mentioned below –

Sr.No.	Variable & Description
1	<b>DISPLAY</b> Contains the identifier for the display that <b>X11</b> programs should use by default.
2	<b>HOME</b> Indicates the home directory of the current user: the default argument for the <b>cd built-in</b> command.
3	<b>IFS</b> Indicates the <b>Internal Field Separator</b> that is used by the parser for word splitting after expansion.
4	<b>LANG</b> LANG expands to the default system locale; LC_ALL can be used to override this. For example, if its value is <b>pt_BR</b> , then the language is set to (Brazilian) Portuguese and the locale to Brazil.
5	<b>LD_LIBRARY_PATH</b> A Unix system with a dynamic linker, contains a colon-separated list of directories that the dynamic linker should search for shared objects when building a process image after exec, before searching in any other directories.
6	<b>PATH</b> Indicates the search path for commands. It is a colon-separated list of directories in which the shell looks for commands.
7	<b>PWD</b> Indicates the current working directory as set by the cd command.

8	<b>RANDOM</b> Generates a random integer between 0 and 32,767 each time it is referenced.
9	<b>SHLVL</b> Increments by one each time an instance of bash is started. This variable is useful for determining whether the built-in exit command ends the current session.
10	<b>TERM</b> Refers to the display type.
11	<b>TZ</b> Refers to Time zone. It can take values like GMT, AST, etc.
12	<b>UID</b> Expands to the numeric user ID of the current user, initialized at the shell startup.

Following is the sample example showing few environment variables –

```
$ echo $HOME
/root
]$ echo $DISPLAY

$ echo $TERM
xterm
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/amrood/bin:/usr/local/bin
$
```

In this chapter, we will discuss in detail about Printing and Email as the basic utilities of Unix. So far, we have tried to understand the Unix OS and the nature of its basic commands. In this chapter, we will learn some important Unix utilities that can be used in our day-to-day life.

## Printing Files

Before you print a file on a Unix system, you may want to reformat it to adjust the margins, highlight some words, and so on. Most files can also be printed without reformatting, but the raw printout may not be that appealing.

Many versions of Unix include two powerful text formatters, **nroff** and **troff**.

### The pr Command

The **pr** command does minor formatting of files on the terminal screen or for a printer. For example, if you have a long list of names in a file, you can format it onscreen into two or more columns.



Following is the syntax for the **pr** command –

```
pr option(s) filename(s)
```

The **pr** changes the format of the file only on the screen or on the printed copy; it doesn't modify the original file. Following table lists some **pr** options –

Sr.No.	Option & Description
1	<b>-k</b> Produces <b>k</b> columns of output
2	<b>-d</b> Double-spaces the output (not on all <b>pr</b> versions)
3	<b>-h "header"</b> Takes the next item as a report header
4	<b>-t</b> Eliminates the printing of header and the top/bottom margins
5	<b>-l PAGE_LENGTH</b> Sets the page length to PAGE_LENGTH (66) lines. The default number of lines of text is 56
6	<b>-o MARGIN</b> Offsets each line with MARGIN (zero) spaces
7	<b>-w PAGE_WIDTH</b> Sets the page width to PAGE_WIDTH (72) characters for multiple text-column output only

Before using **pr**, here are the contents of a sample file named food.

```
$cat food
Sweet Tooth
Bangkok Wok
Mandalay
Afghani Cuisine
Isle of Java
Big Apple Deli
Sushi and Sashimi
Tio Pepe's Peppers
.....
$
```

Let's use the **pr** command to make a two-column report with the header *Restaurants* –

```
$pr -2 -h "Restaurants" food
Nov 7 9:58 1997 Restaurants Page 1

Sweet Tooth           Isle of Java
Bangkok Wok          Big Apple Deli
Mandalay              Sushi and Sashimi
Afghani Cuisine       Tio Pepe's Peppers
.....
$
```

## The lp and lpr Commands

The command **lp** or **lpr** prints a file onto paper as opposed to the screen display. Once you are ready with formatting using the **pr** command, you can use any of these commands to print your file on the printer connected to your computer.

Your system administrator has probably set up a default printer at your site. To print a file named **food** on the default printer, use the **lp** or **lpr** command, as in the following example –

```
$lp food
request id is laserp-525 (1 file)
$
```

The **lp** command shows an ID that you can use to cancel the print job or check its status.

- If you are using the **lp** command, you can use the **-nNum** option to print Num number of copies. Along with the command **lpr**, you can use **-Num** for the same.
- If there are multiple printers connected with the shared network, then you can choose a printer using **-dprinter** option along with lp command and for the same purpose you can use **-Pprinter** option along with lpr command. Here printer is the printer name.

## The lpstat and lpq Commands

The **lpstat** command shows what's in the printer queue: request IDs, owners, file sizes, when the jobs were sent for printing, and the status of the requests.

Use **lpstat -o** if you want to see all output requests other than just your own. Requests are shown in the order they'll be printed –

```
$lpstat -o
laserp-573 john 128865 Nov 7 11:27 on laserp
laserp-574 grace 82744 Nov 7 11:28
laserp-575 john 23347 Nov 7 11:35
$
```

The **lpq** gives slightly different information than **lpstat -o** –

```
$lpq
laserp is ready and printing
Rank Owner Job Files Total Size
active john 573 report.ps 128865 bytes
1st grace 574 ch03.ps ch04.ps 82744 bytes
2nd john 575 standard input 23347 bytes
$
```

Here the first line displays the printer status. If the printer is disabled or running out of paper, you may see different messages on this first line.

## The cancel and lprm Commands

The **cancel** command terminates a printing request from the **lp** command. The **lprm** command terminates all **lpr** requests. You can specify either the ID of the request (displayed by lp or lpq) or the name of the printer.

```
$cancel laserp-575
request "laserp-575" cancelled
$
```

To cancel whatever request is currently printing, regardless of its ID, simply enter cancel and the printer name –

```
$cancel laserp
request "laserp-573" cancelled
$
```

The **lprm** command will cancel the active job if it belongs to you. Otherwise, you can give job numbers as arguments, or use a **dash (-)** to remove all of your jobs –

```
$lprm 575
dfA575diamond dequeued
cfA575diamond dequeued
$
```

The **lprm** command tells you the actual filenames removed from the printer queue.

## Sending Email

You use the Unix mail command to send and receive mail. Here is the syntax to send an email –

```
$mail [-s subject] [-c cc-addr] [-b bcc-addr] to-addr
```

Here are important options related to mail command –

Sr.No.	Option & Description
1	<b>-s</b> Specifies subject on the command line.
2	<b>-c</b> Sends carbon copies to the list of users. List should be a commaseparated list of names.
3	<b>-b</b> Sends blind carbon copies to list. List should be a commaseparated list of names.

Following is an example to send a test message to admin@yahoo.com.

```
$mail -s "Test Message" admin@yahoo.com
```

You are then expected to type in your message, followed by "**control-D**" at the beginning of a line. To stop, simply type dot (**.**) as follows –

```
Hi,
This is a test
.
Cc:
```

You can send a complete file using a **redirect < operator** as follows –

```
$mail -s "Report 05/06/07" admin@yahoo.com < demo.txt
```

To check incoming email at your Unix system, you simply type email as follows –

```
$mail
no email
```

- In this chapter, we will discuss in detail about pipes and filters in Unix. You can connect two commands together so that the output

from one program becomes the input of the next program. Two or more commands connected in this way form a pipe.

- To make a pipe, put a vertical bar (|) on the command line between two commands.
- When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a **filter**.

## • The grep Command

- The grep command searches a file or files for lines that have a certain pattern. The syntax is –

```
• $grep pattern file(s)
```

- The name "**grep**" comes from the ed (a Unix line editor) command **g/re/p** which means "globally search for a regular expression and print all lines containing it".
- A regular expression is either some plain text (a word, for example) and/or special characters used for pattern matching.
- The simplest use of grep is to look for a pattern consisting of a single word. It can be used in a pipe so that only those lines of the input files containing a given string are sent to the standard output. If you don't give grep a filename to read, it reads its standard input; that's the way all filter programs work –

```
• $ls -l | grep "Aug"
• -rw-rw-rw-  1 john  doc    11008 Aug  6 14:10 ch02
• -rw-rw-rw-  1 john  doc     8515 Aug  6 15:30 ch07
• -rw-rw-r--  1 john  doc     2488 Aug 15 10:51 intro
• -rw-rw-r--  1 carol doc     1605 Aug 23 07:35 macros
• $
```

- There are various options which you can use along with the **grep** command –

Sr.No.	Option & Description
1	<b>-v</b> Prints all lines that do not match pattern.
2	<b>-n</b> Prints the matched line and its line number.
3	<b>-l</b> Prints only the names of files with matching lines (letter "l")
4	<b>-c</b> Prints only the count of matching lines.
5	<b>-i</b> Matches either upper or lowercase.

- Let us now use a regular expression that tells grep to find lines with "**carol**", followed by zero or other characters abbreviated in a regular expression as ".\*"), then followed by "Aug".–
- Here, we are using the **-i** option to have case insensitive search –

```
$ls -l | grep -i "carol.*aug"
-rw-rw-r-- 1 carol doc      1605 Aug 23 07:35 macros
$
```

## • The sort Command

- The **sort** command arranges lines of text alphabetically or numerically. The following example sorts the lines in the food file –

```
$sort food
Afghani Cuisine
Bangkok Wok
Big Apple Deli
Isle of Java
Mandalay
Sushi and Sashimi
Sweet Tooth
Tio Pepe's Peppers
$
```

- The **sort** command arranges lines of text alphabetically by default. There are many options that control the sorting –

Sr.No.	Description
1	<b>-n</b> Sorts numerically (example: 10 will sort after 2), ignores blanks and tabs.
2	<b>-r</b> Reverses the order of sort.
3	<b>-f</b> Sorts upper and lowercase together.
4	<b>+x</b> Ignores first <b>x</b> fields when sorting.

- More than two commands may be linked up into a pipe. Taking a previous pipe example using **grep**, we can further sort the files modified in August by the order of size.
- The following pipe consists of the commands **ls**, **grep**, and **sort** –

```
$ls -l | grep "Aug" | sort +4n
-rw-rw-r-- 1 carol doc      1605 Aug 23 07:35 macros
-rw-rw-r-- 1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-rw- 1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-rw- 1 john  doc     11008 Aug  6 14:10 ch02
$
```

- This pipe sorts all files in your directory modified in August by the order of size, and prints them on the terminal screen. The sort option `+4n` skips four fields (fields are separated by blanks) then sorts the lines in numeric order.

## • The `pg` and `more` Commands

- A long output can normally be zipped by you on the screen, but if you run text through `more` or use the `pg` command as a filter; the display stops once the screen is full of text.
- Let's assume that you have a long directory listing. To make it easier to read the sorted listing, pipe the output through **more** as follows –

```

• $ls -l | grep "Aug" | sort +4n | more
• -rw-rw-r-- 1 carol doc      1605 Aug 23 07:35 macros
• -rw-rw-r-- 1 john  doc      2488 Aug 15 10:51 intro
• -rw-rw-rw- 1 john  doc      8515 Aug  6 15:30 ch07
• -rw-rw-r-- 1 john  doc     14827 Aug  9 12:40 ch03
•
•
•
• -rw-rw-rw- 1 john  doc     16867 Aug  6 15:56 ch05
• --More--(74%)

```

- The screen will fill up once the screen is full of text consisting of lines sorted by the order of the file size. At the bottom of the screen is the **more** prompt, where you can type a command to move through the sorted text.
- Once you're done with this screen, you can use any of the commands listed in the discussion of the `more` program.

In this chapter, we will discuss in detail about process management in Unix. When you execute a program on your Unix system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program as if no other program were running on the system.

Whenever you issue a command in Unix, it creates, or starts, a new process. When you tried out the `ls` command to list the directory contents, you started a process. A process, in simple terms, is an instance of a running program.

The operating system tracks processes through a five-digit ID number known as the **pid** or the **process ID**. Each process in the system has a unique **pid**.

Pids eventually repeat because all the possible numbers are used up and the next pid rolls or starts over. At any point of time, no two processes with the same pid exist in the system because it is the pid that Unix uses to track each process.

## Starting a Process

When you start a process (run a command), there are two ways you can run it –

- Foreground Processes
- Background Processes

## Foreground Processes

By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.

You can see this happen with the **ls** command. If you wish to list all the files in your current directory, you can use the following command –

```
$ls ch*.doc
```

This would display all the files, the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc  ch010.doc  ch02.doc  ch03-2.doc
ch04-1.doc  ch040.doc  ch05.doc  ch06-2.doc
ch01-2.doc  ch02-1.doc
```

The process runs in the foreground, the output is directed to my screen, and if the **ls** command wants any input (which it does not), it waits for it from the keyboard.

While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.

## Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (**&**) at the end of the command.

```
$ls ch*.doc &
```

This displays all those files the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc  ch010.doc  ch02.doc  ch03-2.doc
ch04-1.doc  ch040.doc  ch05.doc  ch06-2.doc
ch01-2.doc  ch02-1.doc
```

Here, if the **ls** command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard.

That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground.

Press the Enter key and you will see the following –

```
[1] + Done ls ch*.doc &
$
```

The first line tells you that the **ls** command background process finishes successfully. The second is a prompt for another command.

## Listing Running Processes

It is easy to see your own processes by running the **ps** (process status) command as follows –

```
$ps
PID TTY TIME CMD
18358 ttyp3 00:00:00 sh
18361 ttyp3 00:01:31 abiword
18789 ttyp3 00:00:00 ps
```

One of the most commonly used flags for **ps** is the **-f** ( f for full) option, which provides more information as shown in the following example –

```
$ps -f
UID PID PPID C STIME TTY TIME CMD
amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood 6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood 3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood 6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

Here is the description of all the fields displayed by **ps -f** command –

Sr.No.	Column & Description
1	<b>UID</b> User ID that this process belongs to (the person running it)
2	<b>PID</b> Process ID
3	<b>PPID</b> Parent process ID (the ID of the process that started it)
4	<b>C</b> CPU utilization of process
5	<b>STIME</b> Process start time
6	<b>TTY</b> Terminal type associated with the process
7	<b>TIME</b> CPU time taken by the process
8	<b>CMD</b> The command that started this process



There are other options which can be used along with **ps** command –

Sr.No.	Option & Description
1	<b>-a</b> Shows information about all users
2	<b>-x</b> Shows information about processes without terminals
3	<b>-u</b> Shows additional information like -f option
4	<b>-e</b> Displays extended information

## Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows –

```
$ps -f
UID      PID  PPID  C  STIME  TTY  TIME  CMD
amrood   6738 3662  0  10:23:03 pts/6 0:00 first_one
amrood   6739 3662  0  10:22:54 pts/6 0:00 second_one
amrood   3662 3657  0  08:10:53 pts/6 0:00 -ksh
amrood   6892 3662  4  10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

Here, the **kill** command terminates the **first\_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows –

```
$kill -9 6738
Terminated
```

## Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

## Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

## Daemon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a "**ps -ef**" and look at the **tty** field, all daemons will have a **?** for the **tty**.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then it's worth to make it a daemon and run it in the background.

## The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes –

```
$top
```

## Job ID Versus Process ID

Background and suspended processes are usually manipulated via **job number (job ID)**. This number is different from the process ID and is used because it is shorter.

In addition, a job can consist of multiple processes running in a series or at the same time, in parallel. Using the job ID is easier than tracking individual processes.

In this chapter, we will discuss in detail about network communication utilities in Unix. When you work in a distributed environment, you need to communicate with remote users and you also need to access remote Unix machines.

There are several Unix utilities that help users compute in a networked, distributed environment. This chapter lists a few of them.

## The ping Utility

The **ping** command sends an echo request to a host available on the network. Using this command, you can check if your remote host is responding well or not.

The ping command is useful for the following –

- Tracking and isolating hardware and software problems.
- Determining the status of the network and various foreign hosts.
- Testing, measuring, and managing networks.

## Syntax

Following is the simple syntax to use the ping command –

```
$ping hostname or ip-address
```

The above command starts printing a response after every second. To come out of the command, you can terminate it by pressing **CTRL + C** keys.

## Example

Following is an example to check the availability of a host available on the network –

```
$ping google.com
PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq = 1 ttl = 54 time = 39.4 ms
64 bytes from 74.125.67.100: icmp_seq = 2 ttl = 54 time = 39.9 ms
64 bytes from 74.125.67.100: icmp_seq = 3 ttl = 54 time = 39.3 ms
64 bytes from 74.125.67.100: icmp_seq = 4 ttl = 54 time = 39.1 ms
64 bytes from 74.125.67.100: icmp_seq = 5 ttl = 54 time = 38.8 ms
--- google.com ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 21017ms
rtt min/avg/max/mdev = 38.867/39.334/39.900/0.396 ms
$
```

If a host does not exist, you will receive the following output –

```
$ping giiiiigle.com
ping: unknown host giiiiigle.com
$
```

## The ftp Utility

Here, **ftp** stands for **F**ile **T**ransfer **P**rotocol. This utility helps you upload and download your file from one computer to another computer.

The ftp utility has its own set of Unix-like commands. These commands help you perform tasks such as –

- Connect and login to a remote host.
- Navigate directories.

- List directory contents.
- Put and get files.
- Transfer files as **ascii**, **ebcdic** or **binary**.

## Syntax

Following is the simple syntax to use the ping command –

```
$ftp hostname or ip-address
```

The above command would prompt you for the login ID and the password. Once you are authenticated, you can access the home directory of the login account and you would be able to perform various commands.

The following tables lists out a few important commands –

Sr.No.	Command & Description
1	<b>put filename</b> Uploads filename from the local machine to the remote machine.
2	<b>get filename</b> Downloads filename from the remote machine to the local machine.
3	<b>mput file list</b> Uploads more than one file from the local machine to the remote machine.
4	<b>mget file list</b> Downloads more than one file from the remote machine to the local machine.
5	<b>prompt off</b> Turns the prompt off. By default, you will receive a prompt to upload or download files using <b>mput</b> or <b>mget</b> commands.
6	<b>prompt on</b> Turns the prompt on.
7	<b>dir</b> Lists all the files available in the current directory of the remote machine.
8	<b>cd dirname</b> Changes directory to dirname on the remote machine.
9	<b>lcd dirname</b> Changes directory to dirname on the local machine.
10	<b>quit</b>

Helps logout from the current login.

It should be noted that all the files would be downloaded or uploaded to or from the current directories. If you want to upload your files in a particular directory, you need to first change to that directory and then upload the required files.

## Example

Following is the example to show the working of a few commands –

```
$ftp amrood.com
Connected to amrood.com.
220 amrood.com FTP server (Ver 4.9 Thu Sep 2 20:35:07 CDT 2009)
Name (amrood.com:amrood): amrood
331 Password required for amrood.
Password:
230 User amrood logged in.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls.
total 1464
drwxr-sr-x  3 amrood  group    1024 Mar 11 20:04 Mail
drwxr-sr-x  2 amrood  group    1536 Mar  3 18:07 Misc
drwxr-sr-x  5 amrood  group     512 Dec  7 10:59 OldStuff
drwxr-sr-x  2 amrood  group    1024 Mar 11 15:24 bin
drwxr-sr-x  5 amrood  group    3072 Mar 13 16:10 mpl
-rw-r--r--  1 amrood  group  209671 Mar 15 10:57 myfile.out
drwxr-sr-x  3 amrood  group     512 Jan  5 13:32 public
drwxr-sr-x  3 amrood  group     512 Feb 10 10:17 pvm3
226 Transfer complete.
ftp> cd mpl
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls.
total 7320
-rw-r--r--  1 amrood  group    1630 Aug  8 1994 dboard.f
-rw-r----  1 amrood  group    4340 Jul 17 1994 vttest.c
-rwxr-xr-x  1 amrood  group  525574 Feb 15 11:52 wave_shift
-rw-r--r--  1 amrood  group    1648 Aug  5 1994 wide.list
-rwxr-xr-x  1 amrood  group    4019 Feb 14 16:26 fix.c
226 Transfer complete.
ftp> get wave_shift
200 PORT command successful.
150 Opening data connection for wave_shift (525574 bytes).
226 Transfer complete.
528454 bytes received in 1.296 seconds (398.1 Kbytes/s)
ftp> quit
221 Goodbye.
$
```

## The telnet Utility

There are times when we are required to connect to a remote Unix machine and work on that machine remotely. **Telnet** is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you login using Telnet, you can perform all the activities on your remotely connected machine. The following is an example of Telnet session –

```
C:>telnet amrood.com
Trying...
Connected to amrood.com.
Escape character is '^]'.

login: amrood
amrood's Password:
*****
*
*
*   WELCOME TO AMROOD.COM
*
*
*****

Last unsuccessful login: Fri Mar  3 12:01:09 IST 2009
Last login: Wed Mar  8 18:33:27 IST 2009 on pts/10
```

```
{ do your work }  
$ logout  
Connection closed.  
C:>
```

## The finger Utility

The **finger** command displays information about users on a given host. The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

Following is the simple syntax to use the finger command –

Check all the logged-in users on the local machine –

```
$ finger  
Login Name Tty Idle Login Time Office  
amrood pts/0 Jun 25 08:03 (62.61.164.115)
```

Get information about a specific user available on the local machine –

```
$ finger amrood  
Login: amrood Name: (null)  
Directory: /home/amrood Shell: /bin/bash  
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115  
No mail.  
No Plan.
```

Check all the logged-in users on the remote machine –

```
$ finger @avtar.com  
Login Name Tty Idle Login Time Office  
amrood pts/0 Jun 25 08:03 (62.61.164.115)
```

Get the information about a specific user available on the remote machine –

```
$ finger amrood@avtar.com  
Login: amrood Name: (null)  
Directory: /home/amrood Shell: /bin/bash  
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115  
No mail.  
No Plan.
```

In this chapter, we will understand how the vi Editor works in Unix. There are many ways to edit files in Unix. Editing files using the screen-oriented text editor **vi** is one of the best ways. This editor enables you to edit lines in context with other lines in the file.

An improved version of the vi editor which is called the **VIM** has also been made available now. Here, VIM stands for **Vi IM**proved.

vi is generally considered the de facto standard in Unix editors because –

- It's usually available on all the flavors of Unix system.
- Its implementations are very similar across the board.
- It requires very few resources.
- It is more user-friendly than other editors such as the **ed** or the **ex**.

You can use the **vi** editor to edit an existing file or to create a new file from scratch. You can also use this editor to just read a text file.

## Starting the vi Editor



vi editor. Start by typing some characters and then come to the command mode to understand the difference.

## Getting Out of vi

The command to quit out of vi is **:q**. Once in the command mode, type colon, and 'q', followed by return. If your file has been modified in any way, the editor will warn you of this, and not let you quit. To ignore this message, the command to quit out of vi without saving is **:q!**. This lets you exit vi without saving any of the changes.

The command to save the contents of the editor is **:w**. You can combine the above command with the quit command, or use **:wq** and return.

The easiest way to **save your changes and exit vi** is with the ZZ command. When you are in the command mode, type **ZZ**. The **ZZ** command works the same way as the **:wq** command.

If you want to specify/state any particular name for the file, you can do so by specifying it after the **:w**. For example, if you wanted to save the file you were working on as another filename called **filename2**, you would type **:w filename2** and return.

## Moving within a File

To move around within a file without affecting your text, you must be in the command mode (press Esc twice). The following table lists out a few commands you can use to move around one character at a time –

Sr.No.	Command & Description
1	<b>k</b> Moves the cursor up one line
2	<b>j</b> Moves the cursor down one line
3	<b>h</b> Moves the cursor to the left one character position
4	<b>l</b> Moves the cursor to the right one character position

The following points need to be considered to move within a file –

- vi is case-sensitive. You need to pay attention to capitalization when using the commands.
- Most commands in vi can be prefaced by the number of times you want the action to occur. For example, **2j** moves the cursor two lines down the cursor location.



There are many other ways to move within a file in vi. Remember that you must be in the command mode (**press Esc twice**). The following table lists out a few commands to move around the file –

Given below is the list of commands to move around the file.

## Control Commands

The following commands can be used with the Control Key to performs functions as given in the table below –

Given below is the list of control commands.

## Editing Files

To edit the file, you need to be in the insert mode. There are many ways to enter the insert mode from the command mode –

Sr.No.	Command & Description
1	<b>i</b> Inserts text before the current cursor location
2	<b>I</b> Inserts text at the beginning of the current line
3	<b>a</b> Inserts text after the current cursor location
4	<b>A</b> Inserts text at the end of the current line
5	<b>o</b> Creates a new line for text entry below the cursor location
6	<b>O</b> Creates a new line for text entry above the cursor location

## Deleting Characters

Here is a list of important commands, which can be used to delete characters and lines in an open file –

Sr.No.	Command & Description
1	<b>x</b> Deletes the character under the cursor location
2	<b>X</b>

	Deletes the character before the cursor location
3	<b>dw</b> Deletes from the current cursor location to the next word
4	<b>d^</b> Deletes from the current cursor position to the beginning of the line
5	<b>d\$</b> Deletes from the current cursor position to the end of the line
6	<b>D</b> Deletes from the cursor position to the end of the current line
7	<b>dd</b> Deletes the line the cursor is on

As mentioned above, most commands in vi can be prefaced by the number of times you want the action to occur. For example, **2x** deletes two characters under the cursor location and **2dd** deletes two lines the cursor is on.

It is recommended that the commands are practiced before we proceed further.

## Change Commands

You also have the capability to change characters, words, or lines in vi without deleting them. Here are the relevant commands –

Sr.No.	Command & Description
1	<b>cc</b> Removes the contents of the line, leaving you in insert mode.
2	<b>cw</b> Changes the word the cursor is on from the cursor to the lowercase <b>w</b> end of the word.
3	<b>r</b> Replaces the character under the cursor. vi returns to the command mode after the replacement is entered.
4	<b>R</b> Overwrites multiple characters beginning with the character currently under the cursor. You must use <b>Esc</b> to stop the overwriting.
5	<b>s</b>

	Replaces the current character with the character you type. Afterward, you are left in the insert mode.
6	<b>s</b>  Deletes the line the cursor is on and replaces it with the new text. After the new text is entered, vi remains in the insert mode.

## Copy and Paste Commands

You can copy lines or words from one place and then you can paste them at another place using the following commands –

Sr.No.	Command & Description
1	<b>yy</b>  Copies the current line.
2	<b>yw</b>  Copies the current word from the character the lowercase w cursor is on, until the end of the word.
3	<b>p</b>  Puts the copied text after the cursor.
4	<b>P</b>  Puts the yanked text before the cursor.

## Advanced Commands

There are some advanced commands that simplify day-to-day editing and allow for more efficient use of vi –

Given below is the list advanced commands.

### Word and Character Searching

The vi editor has two kinds of searches: **string** and **character**. For a string search, the **/** and **?** commands are used. When you start these commands, the command just typed will be shown on the last line of the screen, where you type the particular string to look for.

These two commands differ only in the direction where the search takes place –

- The **/** command searches forwards (downwards) in the file.
- The **?** command searches backwards (upwards) in the file.

The **n** and **N** commands repeat the previous search command in the same or the opposite direction, respectively. Some characters have

special meanings. These characters must be preceded by a backslash (\) to be included as part of the search expression.

Sr.No.	Character & Description
1	^ Searches at the beginning of the line (Use at the beginning of a search expression).
2	. Matches a single character.
3	* Matches zero or more of the previous character.
4	\$ End of the line (Use at the end of the search expression).
5	[ Starts a set of matching or non-matching expressions.
6	< This is put in an expression escaped with the backslash to find the ending or the beginning of a word.
7	> This helps see the '<' character description above.

The character search searches within one line to find a character entered after the command. The **f** and **F** commands search for a character on the current line only. **f** searches forwards and **F** searches backwards and the cursor moves to the position of the found character.

The **t** and **T** commands search for a character on the current line only, but for **t**, the cursor moves to the position before the character, and **T** searches the line backwards to the position after the character.

## Set Commands

You can change the look and feel of your vi screen using the following **:set** commands. Once you are in the command mode, type **:set** followed by any of the following commands.

Sr.No.	Command & Description
1	<b>:set ic</b> Ignores the case when searching

2	<b>:set ai</b> Sets autoindent
3	<b>:set noai</b> Unsets autoindent
4	<b>:set nu</b> Displays lines with line numbers on the left side
5	<b>:set sw</b> Sets the width of a software tabstop. For example, you would set a shift width of 4 with this command — <b>:set sw = 4</b>
6	<b>:set ws</b> If <i>wraps</i> is set, and the word is not found at the bottom of the file, it will try searching for it at the beginning
7	<b>:set wm</b> If this option has a value greater than zero, the editor will automatically "word wrap". For example, to set the wrap margin to two characters, you would type this: <b>:set wm = 2</b>
8	<b>:set ro</b> Changes file type to "read only"
9	<b>:set term</b> Prints terminal type
10	<b>:set bf</b> Discards control characters from input

## Running Commands

The vi has the capability to run commands from within the editor. To run a command, you only need to go to the command mode and type **:! command**.

For example, if you want to check whether a file exists before you try to save your file with that filename, you can type **:! ls** and you will see the output of **ls** on the screen.

You can press any key (or the command's escape sequence) to return to your vi session.

## Replacing Text

The substitution command (**:s/**) enables you to quickly replace words or groups of words within your files. Following is the syntax to replace text –

```
:s/search/replace/g
```

The **g** stands for globally. The result of this command is that all occurrences on the cursor's line are changed.

## Important Points to Note

The following points will add to your success with vi –

- You must be in command mode to use the commands. (Press Esc twice at any time to ensure that you are in command mode.)
- You must be careful with the commands. These are case-sensitive.
- You must be in insert mode to enter text.

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

## Extended Shell Scripts

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.

The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

The following script uses the **read** command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

```
#!/bin/sh

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:

echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Here is a sample run of the script –

```
./test.sh
What is your name?
Zara Ali
Hello, Zara Ali
$
```

Subsequent part of this tutorial will cover Unix/Linux Shell Scripting in detail.

A **Shell** provides you with an interface to the Unix system. 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 Prompt

The prompt, **\$**, which is called the **command prompt**, is issued by the shell. While the prompt is displayed, you can type a command.

Shell reads your input after you press **Enter**. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Following is a simple example of the **date** command, which displays the current date and time –

```
$date
Thu Jun 25 08:30:19 MST 2009
```

You can customize your command prompt using the environment variable PS1 explained in the Environment tutorial.

## Shell Types

In Unix, 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.

The Bourne Shell has the following subcategories –

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

The different C-type shells follow –

- C shell (csh)
- TENEX/TOPS C shell (tcsh)

The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.

Bourne shell was the first shell to appear on Unix systems, thus it is referred to as "the shell".

Bourne shell is usually installed as **/bin/sh** on most versions of Unix. For this reason, it is the shell of choice for writing scripts that can be used on different versions of Unix.

In this chapter, we are going to cover most of the Shell concepts that are based on the Bourne Shell.

## Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by **#** sign, describing the steps.

There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.

We are going to write many scripts in the next sections. It would be a simple text file in which we would put all our commands and several other required constructs that tell the shell environment what to do and when to do it.

Shell scripts and functions are both interpreted. This means they are not compiled.

## Example Script

Assume we create a **test.sh** script. Note all the scripts would have the **.sh** extension. Before you add anything else to your script, you need to alert the system that a shell script is being started. This is done using the **shebang** construct. For example –

```
#!/bin/sh
```

This tells the system that the commands that follow are to be executed by the Bourne shell. *It's called a shebang because the # symbol is called a hash, and the ! symbol is called a bang.*

To create a script containing these commands, you put the shebang line first and then add the commands –

```
#!/bin/bash
pwd
ls
```

## Shell Comments

You can put your comments in your script as follows –

```
#!/bin/bash
```



```
# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:
pwd
ls
```

Save the above content and make the script executable –

```
$chmod +x test.sh
```

The shell script is now ready to be executed –

```
$/test.sh
```

Upon execution, you will receive the following result –

```
/home/amrood
index.htm  unix-basic_utilities.htm  unix-directories.htm
test.sh   unix-communication.htm   unix-environment.htm
```

**Note** – To execute a program available in the current directory, use **./program\_name**

In this chapter, we will understand shell decision-making in Unix. While writing a shell script, there may be a situation when you need to adopt one path out of the given two paths. So you need to make use of conditional statements that allow your program to make correct decisions and perform the right actions.

Unix Shell supports conditional statements which are used to perform different actions based on different conditions. We will now understand two decision-making statements here –

- The **if...else** statement
- The **case...esac** statement

## The if...else statements

If else statements are useful decision-making statements which can be used to select an option from a given set of options.

Unix Shell supports following forms of **if...else** statement –

- if...fi statement
- if...else...fi statement
- if...elif...else...fi statement

Most of the if statements check relations using relational operators discussed in the previous chapter.

## The case...esac Statement

You can use multiple **if...elif** statements to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Unix Shell supports **case...esac** statement which handles exactly this situation, and it does so more efficiently than repeated **if...elif** statements.

There is only one form of **case...esac** statement which has been described in detail here –

- `case...esac` statement

The **case...esac** statement in the Unix shell is very similar to the **switch...case** statement we have in other programming languages like **C** or **C++** and **PERL**, etc.

In this chapter, we will discuss shell loops in Unix. A loop is a powerful programming tool that enables you to execute a set of commands repeatedly. In this chapter, we will examine the following types of loops available to shell programmers –

- [The while loop](#)
- [The for loop](#)
- [The until loop](#)
- [The select loop](#)

You will use different loops based on the situation. For example, the **while** loop executes the given commands until the given condition remains true; the **until** loop executes until a given condition becomes true.

Once you have good programming practice you will gain the expertise and thereby, start using appropriate loop based on the situation. Here, **while** and **for** loops are available in most of the other programming languages like **C**, **C++** and **PERL**, etc.

## Nesting Loops

All the loops support nesting concept which means you can put one loop inside another similar one or different loops. This nesting can go up to unlimited number of times based on your requirement.

Here is an example of nesting **while** loop. The other loops can be nested based on the programming requirement in a similar way –

## Nesting while Loops

It is possible to use a while loop as part of the body of another while loop.

### Syntax

```
while command1 ; # this is loop1, the outer loop
do
    Statement(s) to be executed if command1 is true

    while command2 ; # this is loop2, the inner loop
    do
        Statement(s) to be executed if command2 is true
    done

    Statement(s) to be executed if command1 is true
done
```

### Example

Here is a simple example of loop nesting. Let's add another countdown loop inside the loop that you used to count to nine –

```
#!/bin/sh

a=0
while [ "$a" -lt 10 ] # this is loop1
do
    b="$a"
    while [ "$b" -ge 0 ] # this is loop2
    do
        echo -n "$b "
        b=`expr $b - 1`
    done
    echo
    a=`expr $a + 1`
done
```

This will produce the following result. It is important to note how **echo -n** works here. Here **-n** option lets echo avoid printing a new line character.

```
0
1 0
2 1 0
3 2 1 0
4 3 2 1 0
5 4 3 2 1 0
6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
8 7 6 5 4 3 2 1 0
9 8 7 6 5 4 3 2 1 0
```

In this chapter, we will discuss shell loop control in Unix. So far you have looked at creating loops and working with loops to accomplish different tasks. Sometimes you need to stop a loop or skip iterations of the loop.

In this chapter, we will learn following two statements that are used to control shell loops—

- The **break** statement
- The **continue** statement

## The infinite Loop

All the loops have a limited life and they come out once the condition is false or true depending on the loop.

A loop may continue forever if the required condition is not met. A loop that executes forever without terminating executes for an infinite number of times. For this reason, such loops are called infinite loops.

### Example

Here is a simple example that uses the **while** loop to display the numbers zero to nine –

```
#!/bin/sh
```

```
a=10

until [ $a -lt 10 ]
do
    echo $a
    a=expr $a + 1`
done
```

This loop continues forever because **a** is always **greater than** or **equal to 10** and it is never less than 10.

## The break Statement

The **break** statement is used to terminate the execution of the entire loop, after completing the execution of all of the lines of code up to the break statement. It then steps down to the code following the end of the loop.

### Syntax

The following **break** statement is used to come out of a loop –

```
break
```

The break command can also be used to exit from a nested loop using this format –

```
break n
```

Here **n** specifies the **n<sup>th</sup>** enclosing loop to the exit from.

### Example

Here is a simple example which shows that loop terminates as soon as **a** becomes 5 –

```
#!/bin/sh

a=0

while [ $a -lt 10 ]
do
    echo $a
    if [ $a -eq 5 ]
    then
        break
    fi
    a=`expr $a + 1`
done
```

Upon execution, you will receive the following result –

```
0
1
2
3
4
```

Here is a simple example of nested for loop. This script breaks out of both loops if **var1 equals 2** and **var2 equals 0** –

[Live Demo](#)

```
#!/bin/sh

for var1 in 1 2 3
do
  for var2 in 0 5
  do
    if [ $var1 -eq 2 -a $var2 -eq 0 ]
    then
      break 2
    else
      echo "$var1 $var2"
    fi
  done
done
```

Upon execution, you will receive the following result. In the inner loop, you have a break command with the argument 2. This indicates that if a condition is met you should break out of outer loop and ultimately from the inner loop as well.

```
1 0
1 5
```

## The continue statement

The **continue** statement is similar to the **break** command, except that it causes the current iteration of the loop to exit, rather than the entire loop.

This statement is useful when an error has occurred but you want to try to execute the next iteration of the loop.

## Syntax

```
continue
```

Like with the break statement, an integer argument can be given to the continue command to skip commands from nested loops.

```
continue n
```

Here **n** specifies the **n<sup>th</sup>** enclosing loop to continue from.

## Example

The following loop makes use of the **continue** statement which returns from the continue statement and starts processing the next statement –

[Live Demo](#)

```
#!/bin/sh
```

```
NUMS="1 2 3 4 5 6 7"

for NUM in $NUMS
do
    Q=`expr $NUM % 2`
    if [ $Q -eq 0 ]
    then
        echo "Number is an even number!!"
        continue
    fi
    echo "Found odd number"
done
```

Upon execution, you will receive the following result –

```
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
```

## Unix / Linux - User Administration

Advertisements

---

[Previous Page](#)

[Next Page](#)

---

In this chapter, we will discuss in detail about user administration in Unix.

There are three types of accounts on a Unix system –

### Root account

This is also called **superuser** and would have complete and unfettered control of the system. A superuser can run any commands without any restriction. This user should be assumed as a system administrator.

### System accounts

System accounts are those needed for the operation of system-specific components for example mail accounts and the **sshd** accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.

## User accounts

User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Unix supports a concept of *Group Account* which logically groups a number of accounts. Every account would be a part of another group account. A Unix group plays important role in handling file permissions and process management.

## Managing Users and Groups

There are four main user administration files –

- [/etc/passwd](#) – Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- [/etc/shadow](#) – Holds the encrypted password of the corresponding account. Not all the systems support this file.
- [/etc/group](#) – This file contains the group information for each account.
- [/etc/gshadow](#) – This file contains secure group account information.

Check all the above files using the **cat** command.

The following table lists out commands that are available on majority of Unix systems to create and manage accounts and groups –

Sr.No.	Command & Description
1	<b>useradd</b> Adds accounts to the system
2	<b>usermod</b> Modifies account attributes
3	<b>userdel</b> Deletes accounts from the system
4	<b>groupadd</b> Adds groups to the system
5	<b>groupmod</b> Modifies group attributes
6	<b>groupdel</b> Removes groups from the system

You can use [Manpage Help](#) to check complete syntax for each command mentioned here.

## Create a Group

We will now understand how to create a group. For this, we need to create groups before creating any account otherwise, we can make use of the existing groups in our system. We have all the groups listed in **/etc/groups** file.

All the default groups are system account specific groups and it is not recommended to use them for ordinary accounts. So, following is the syntax to create a new group account –

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

The following table lists out the parameters –

Sr.No.	Option & Description
1	<b>-g GID</b> The numerical value of the group's ID
2	<b>-o</b> This option permits to add group with non-unique GID
3	<b>-r</b> This flag instructs <b>groupadd</b> to add a system account
4	<b>-f</b> This option causes to just exit with success status, if the specified group already exists. With -g, if the specified GID already exists, other (unique) GID is chosen
5	<b>groupname</b> Actual group name to be created

If you do not specify any parameter, then the system makes use of the default values.

Following example creates a *developers* group with default values, which is very much acceptable for most of the administrators.

```
$ groupadd developers
```

## Modify a Group

To modify a group, use the **groupmod** syntax –

```
groupmod -n new modified group name old group name
```

To change the *developers 2* group name to *developer*, type –

```
$ groupmod -n developer developer 2
```

Here is how you will change the financial GID to 545 –

```
$ groupmod -g 545 developer
```



## Delete a Group

We will now understand how to delete a group. To delete an existing group, all you need is the **groupdel command** and the **group name**. To delete the financial group, the command is –

```
$ groupdel developer
```

This removes only the group, not the files associated with that group. The files are still accessible by their owners.

## Create an Account

Let us see how to create a new account on your Unix system. Following is the syntax to create a user's account –

```
useradd -d homedir -g groupname -m -s shell -u userid accountname
```

The following table lists out the parameters –

Sr.No.	Option & Description
1	<b>-d homedir</b> Specifies home directory for the account
2	<b>-g groupname</b> Specifies a group account for this account
3	<b>-m</b> Creates the home directory if it doesn't exist
4	<b>-s shell</b> Specifies the default shell for this account
5	<b>-u userid</b> You can specify a user id for this account
6	<b>accountname</b> Actual account name to be created

If you do not specify any parameter, then the system makes use of the default values. The **useradd** command modifies the **/etc/passwd**, **/etc/shadow**, and **/etc/group** files and creates a home directory.

Following is the example that creates an account **mcmohd**, setting its home directory to **/home/mcmohd** and the group as **developers**. This user would have Korn Shell assigned to it.

```
$ useradd -d /home/mcmohd -g developers -s /bin/ksh mcmohd
```

Before issuing the above command, make sure you already have the *developers* group created using the **groupadd** command.

Once an account is created you can set its password using the **passwd** command as follows –

```
$ passwd mcmohd20
Changing password for user mcmohd20.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

When you type **passwd accountname**, it gives you an option to change the password, provided you are a superuser. Otherwise, you can change just your password using the same command but without specifying your account name.

## Modify an Account

The **usermod** command enables you to make changes to an existing account from the command line. It uses the same arguments as the **useradd** command, plus the **-l** argument, which allows you to change the account name.

For example, to change the account name **mcmohd** to **mcmohd20** and to change home directory accordingly, you will need to issue the following command –

```
$ usermod -d /home/mcmohd20 -m -l mcmohd mcmohd20
```

## Delete an Account

The **userdel** command can be used to delete an existing user. This is a very dangerous command if not used with caution.

There is only one argument or option available for the command **.r**, for removing the account's home directory and mail file.

For example, to remove account **mcmohd20**, issue the following command –

```
$ userdel -r mcmohd20
```

If you want to keep the home directory for backup purposes, omit the **-r** option. You can remove the home directory as needed at a later time.

---

In this chapter, we will discuss in detail about the Shell quoting mechanisms. We will start by discussing the metacharacters.

## The Metacharacters

Unix Shell provides various metacharacters which have special meaning while using them in any Shell Script and causes termination of a word unless quoted.

For example, `?` matches with a single character while listing files in a directory and an `*` matches more than one character. Here is a list of most of the shell special characters (also called metacharacters) –

```
* ? [ ] ' " \ $ ; & ( ) | ^ < > new-line space tab
```

A character may be quoted (i.e., made to stand for itself) by preceding it with a `\`.

## Example

Following example shows how to print a `*` or a `?` –

[Live Demo](#)

```
#!/bin/sh

echo Hello; Word
```

Upon execution, you will receive the following result –

```
Hello
./test.sh: line 2: Word: command not found
shell returned 127
```

Let us now try using a quoted character –

[Live Demo](#)

```
#!/bin/sh

echo Hello\; Word
```

Upon execution, you will receive the following result –

```
Hello; Word
```

The `$` sign is one of the metacharacters, so it must be quoted to avoid special handling by the shell –

[Live Demo](#)

```
#!/bin/sh

echo "I have \$1200"
```

Upon execution, you will receive the following result –

```
I have $1200
```

The following table lists the four forms of quoting –

Sr.No.	Quoting & Description
1	<b>Single quote</b> All special characters between these quotes lose their special meaning.
2	<b>Double quote</b> Most special characters between these quotes lose their special meaning with these

	<p>exceptions –</p> <ul style="list-style-type: none"> <li>• \$</li> <li>• `</li> <li>• \</li> <li>• \'</li> <li>• \"</li> <li>• \\</li> </ul>
3	<p><b>Backslash</b></p> <p>Any character immediately following the backslash loses its special meaning.</p>
4	<p><b>Back quote</b></p> <p>Anything in between back quotes would be treated as a command and would be executed.</p>

## The Single Quotes

Consider an echo command that contains many special shell characters –

```
echo <-$1500.**>; (update?) [y|n]
```

Putting a backslash in front of each special character is tedious and makes the line difficult to read –

```
echo \<-\$1500.\*\>\; \(\update\?\) \[y|n\]
```

There is an easy way to quote a large group of characters. Put a single quote (') at the beginning and at the end of the string –

```
echo '<-$1500.**>; (update?) [y|n]'
```

Characters within single quotes are quoted just as if a backslash is in front of each character. With this, the echo command displays in a proper way.

If a single quote appears within a string to be output, you should not put the whole string within single quotes instead you should precede that using a backslash (\) as follows –

```
echo 'It\'s Shell Programming'
```

## The Double Quotes

Try to execute the following shell script. This shell script makes use of single quote –

[Live Demo](#)

```
VAR=ZARA
```

```
echo '$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]'
```

Upon execution, you will receive the following result –

```
$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]
```

This is not what had to be displayed. It is obvious that single quotes prevent variable substitution. If you want to substitute variable values and to make inverted commas work as expected, then you would need to put your commands in double quotes as follows –

[Live Demo](#)

```
VAR=ZARA
echo "$VAR owes <-\$1500.**>; [ as of (`date +%m/%d`) ]"
```

Upon execution, you will receive the following result –

```
ZARA owes <-\$1500.**>; [ as of (07/02) ]
```

Double quotes take away the special meaning of all characters except the following –

- `$` for parameter substitution
- Backquotes for command substitution
- `\$` to enable literal dollar signs
- `\`` to enable literal backquotes
- `\"` to enable embedded double quotes
- `\\` to enable embedded backslashes
- All other `\` characters are literal (not special)

Characters within single quotes are quoted just as if a backslash is in front of each character. This helps the echo command display properly.

If a single quote appears within a string to be output, you should not put the whole string within single quotes instead you should precede that using a backslash (`\`) as follows –

```
echo 'It\'s Shell Programming'
```

## The Backquotes

Putting any Shell command in between **backquotes** executes the command.

### Syntax

Here is the simple syntax to put any Shell **command** in between backquotes –

```
var=`command`
```

### Example

The **date** command is executed in the following example and the produced result is stored in DATA variable.

[Live Demo](#)

```
DATE=`date`
```

```
echo "Current Date: $DATE"
```

Upon execution, you will receive the following result –

```
Current Date: Thu Jul 2 05:28:45 MST 2009
```

In this chapter, we will discuss in detail about the system performance in Unix.

We will introduce you to a few free tools that are available to monitor and manage performance on Unix systems. These tools also provide guidelines on how to diagnose and fix performance problems in the Unix environment.

Unix has following major resource types that need to be monitored and tuned –

- **CPU**
- **Memory**
- **Disk space**
- **Communications lines**
- **I/O Time**
- **Network Time**
- **Applications programs**

## Performance Components

The following table lists out five major components which take up the system time –

Sr.No.	Component & Description
1	<b>User State CPU</b> The actual amount of time the CPU spends running the users' program in the user state. It includes the time spent executing library calls, but does not include the time spent in the kernel on its behalf
2	<b>System State CPU</b> This is the amount of time the CPU spends in the system state on behalf of this program. All <b>I/O routines</b> require kernel services. The programmer can affect this value by blocking I/O transfers
3	<b>I/O Time and Network Time</b> This is the amount of time spent moving data and servicing I/O requests
4	<b>Virtual Memory Performance</b> This includes context switching and swapping

5	<p><b>Application Program</b></p> <p>Time spent running other programs - when the system is not servicing this application because another application currently has the CPU</p>
---	--

## Performance Tools

Unix provides following important tools to measure and fine tune Unix system performance –

Sr.No.	Command & Description
1	<p><b>nice/renice</b></p> <p>Runs a program with modified scheduling priority</p>
2	<p><b>netstat</b></p> <p>Prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships</p>
3	<p><b>time</b></p> <p>Helps time a simple command or give resource usage</p>
4	<p><b>uptime</b></p> <p>This is System Load Average</p>
5	<p><b>ps</b></p> <p>Reports a snapshot of the current processes</p>
6	<p><b>vmstat</b></p> <p>Reports virtual memory statistics</p>
7	<p><b>gprof</b></p> <p>Displays call graph profile data</p>
8	<p><b>prof</b></p> <p>Facilitates Process Profiling</p>
9	<p><b>top</b></p> <p>Displays system tasks</p>

Yo

## Extended Shell Scripts

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.

The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

The following script uses the **read** command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

```
#!/bin/sh

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:

echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Here is a sample run of the script –

```
./test.sh
What is your name?
Zara Ali
Hello, Zara Ali
$
```

There are various operators supported by each shell. We will discuss in detail about Bourne shell (default shell) in this chapter.

We will now discuss the following operators –

- Arithmetic Operators
- Relational Operators
- Boolean Operators
- String Operators
- File Test Operators

Bourne shell didn't originally have any mechanism to perform simple arithmetic operations but it uses external programs, either **awk** or **expr**.

The following example shows how to add two numbers –

```
#!/bin/sh

val=`expr 2 + 2`
echo "Total value : $val"
```

The above script will generate the following result –

```
Total value : 4
```



The following points need to be considered while adding –

- There must be spaces between operators and expressions. For example, 2+2 is not correct; it should be written as 2 + 2.
- The complete expression should be enclosed between ```, called the backtick.

## Arithmetic Operators

The following arithmetic operators are supported by Bourne Shell.

Assume variable **a** holds 10 and variable **b** holds 20 then –

### Show Examples

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator	`expr \$a + \$b` will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand	`expr \$a - \$b` will give -10
* (Multiplication)	Multiplies values on either side of the operator	`expr \$a \* \$b` will give 200
/ (Division)	Divides left hand operand by right hand operand	`expr \$b / \$a` will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr \$b % \$a` will give 0
= (Assignment)	Assigns right operand in left operand	a = \$b would assign value of b into a
== (Equality)	Compares two numbers, if both are same then returns true.	[ \$a == \$b ] would return false.
!= (Not Equality)	Compares two numbers, if both are different then returns true.	[ \$a != \$b ] would return true.

It is very important to understand that all the conditional expressions should be inside square braces with spaces around them, for example [ **\$a == \$b** ] is correct whereas, [**\$a==\$b**] is incorrect.

All the arithmetical calculations are done using long integers.

## Relational Operators

Bourne Shell supports the following relational operators that are specific to numeric values. These operators do not work for string values unless their value is numeric.

For example, following operators will work to check a relation between 10 and 20 as well as in between "10" and "20" but not in between "ten" and "twenty".

Assume variable **a** holds 10 and variable **b** holds 20 then –

## Show Examples

Operator	Description	Example
<b>-eq</b>	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[ \$a -eq \$b ] is not true.
<b>-ne</b>	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[ \$a -ne \$b ] is true.
<b>-gt</b>	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[ \$a -gt \$b ] is not true.
<b>-lt</b>	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[ \$a -lt \$b ] is true.
<b>-ge</b>	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[ \$a -ge \$b ] is not true.
<b>-le</b>	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[ \$a -le \$b ] is true.

It is very important to understand that all the conditional expressions should be placed inside square braces with spaces around them. For example, [ **\$a <= \$b** ] is correct whereas, [**\$a <= \$b**] is incorrect.

## Boolean Operators

The following Boolean operators are supported by the Bourne Shell.

Assume variable **a** holds 10 and variable **b** holds 20 then –

## Show Examples

Operator	Description	Example
<b>!</b>	This is logical negation. This inverts a true condition into false and vice versa.	[ ! false ] is true.
<b>-o</b>	This is logical <b>OR</b> . If one of the operands is true, then the condition becomes true.	[ \$a -lt 20 -o \$b -gt 100 ] is true.
<b>-a</b>	This is logical <b>AND</b> . If both the operands are true, then the condition becomes true otherwise false.	[ \$a -lt 20 -a \$b -gt 100 ] is false.

## String Operators

The following string operators are supported by Bourne Shell.

Assume variable **a** holds "abc" and variable **b** holds "efg" then –

## Show Examples

Operator	Description	Example
----------	-------------	---------

<b>=</b>	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[ \$a = \$b ] is not true.
<b>!=</b>	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[ \$a != \$b ] is true.
<b>-z</b>	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[ -z \$a ] is not true.
<b>-n</b>	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[ -n \$a ] is not false.
<b>Str</b>	Checks if <b>str</b> is not the empty string; if it is empty, then it returns false.	[ \$a ] is not false.

## File Test Operators

We have a few operators that can be used to test various properties associated with a Unix file.

Assume a variable **file** holds an existing file name "test" the size of which is 100 bytes and has **read**, **write** and **execute** permission on –

Show Examples

Operator	Description	Example
<b>-b file</b>	Checks if file is a block special file; if yes, then the condition becomes true.	[ -b \$file ] is false.
<b>-c file</b>	Checks if file is a character special file; if yes, then the condition becomes true.	[ -c \$file ] is false.
<b>-d file</b>	Checks if file is a directory; if yes, then the condition becomes true.	[ -d \$file ] is not true.
<b>-f file</b>	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[ -f \$file ] is true.
<b>-g file</b>	Checks if file has its set group ID (SGID) bit set; if yes, then the condition becomes true.	[ -g \$file ] is false.
<b>-k file</b>	Checks if file has its sticky bit set; if yes, then the condition becomes true.	[ -k \$file ] is false.
<b>-p file</b>	Checks if file is a named pipe; if yes, then the condition becomes true.	[ -p \$file ] is false.
<b>-t file</b>	Checks if file descriptor is open and associated with a terminal; if yes, then the condition becomes true.	[ -t \$file ] is false.
<b>-u file</b>	Checks if file has its Set User ID (SUID) bit set; if yes, then the condition becomes true.	[ -u \$file ] is false.
<b>-r file</b>	Checks if file is readable; if yes, then the condition becomes true.	[ -r \$file ] is true.

<b>-w file</b>	Checks if file is writable; if yes, then the condition becomes true.	[ -w \$file ] is true.
<b>-x file</b>	Checks if file is executable; if yes, then the condition becomes true.	[ -x \$file ] is true.
<b>-s file</b>	Checks if file has size greater than 0; if yes, then condition becomes true.	[ -s \$file ] is true.
<b>-e file</b>	Checks if file exists; is true even if file is a directory but exists.	[ -e \$file ] is true.

## C Shell Operators

Following link will give you a brief idea on C Shell Operators –

[C Shell Operators](#)

## Korn Shell Operators

Following link helps you understand Korn Shell Operators –

[Korn Shell Operators](#)

Input output redirection

In this chapter, we will discuss in detail about the Shell input/output redirections. Most Unix system commands take input from your terminal and send the resulting output back to your terminal. A command normally reads its input from the standard input, which happens to be your terminal by default. Similarly, a command normally writes its output to standard output, which is again your terminal by default.

## Output Redirection

The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection.

If the notation `> file` is appended to any command that normally writes its output to standard output, the output of that command will be written to file instead of your terminal.

Check the following **who** command which redirects the complete output of the command in the users file.

```
$ who > users
```

Notice that no output appears at the terminal. This is because the output has been redirected from the default standard output device (the terminal) into the specified file. You can check the users file for the complete content –

```
$ cat users
oko      tty01   Sep 12 07:30
ai       tty15   Sep 12 13:32
ruth     tty21   Sep 12 10:10
pat      tty24   Sep 12 13:07
steve    tty25   Sep 12 13:03
```

```
$
```

If a command has its output redirected to a file and the file already contains some data, that data will be lost. Consider the following example –

```
$ echo line 1 > users
$ cat users
line 1
$
```

You can use `>>` operator to append the output in an existing file as follows –

```
$ echo line 2 >> users
$ cat users
line 1
line 2
$
```

## Input Redirection

Just as the output of a command can be redirected to a file, so can the input of a command be redirected from a file. As the **greater-than character** `>` is used for output redirection, the **less-than character** `<` is used to redirect the input of a command.

The commands that normally take their input from the standard input can have their input redirected from a file in this manner. For example, to count the number of lines in the file *users* generated above, you can execute the command as follows –

```
$ wc -l users
2 users
$
```

Upon execution, you will receive the following output. You can count the number of lines in the file by redirecting the standard input of the **wc** command from the file *users* –

```
$ wc -l < users
2
$
```

Note that there is a difference in the output produced by the two forms of the `wc` command. In the first case, the name of the file *users* is listed with the line count; in the second case, it is not.

In the first case, `wc` knows that it is reading its input from the file *users*. In the second case, it only knows that it is reading its input from standard input so it does not display file name.

## Here Document

A **here document** is used to redirect input into an interactive shell script or program.

We can run an interactive program within a shell script without user action by supplying the required input for the interactive program, or interactive shell script.

The general form for a **here** document is –

```
command << delimiter
```

```
document
delimiter
```

Here the shell interprets the `<<` operator as an instruction to read input until it finds a line containing the specified delimiter. All the input lines up to the line containing the delimiter are then fed into the standard input of the command.

The delimiter tells the shell that the **here** document has completed. Without it, the shell continues to read the input forever. The delimiter must be a single word that does not contain spaces or tabs.

Following is the input to the command **wc -l** to count the total number of lines –

```
$wc -l << EOF
  This is a simple lookup program
    for good (and bad) restaurants
    in Cape Town.
EOF
3
$
```

You can use the **here document** to print multiple lines using your script as follows –

```
#!/bin/sh

cat << EOF
This is a simple lookup program
for good (and bad) restaurants
in Cape Town.
EOF
```

Upon execution, you will receive the following result –

```
This is a simple lookup program
for good (and bad) restaurants
in Cape Town.
```

The following script runs a session with the **vi** text editor and saves the input in the file **test.txt**.

```
#!/bin/sh

filename=test.txt
vi $filename <<EndOfCommands
i
This file was created automatically from
a shell script
^[
ZZ
EndOfCommands
```

If you run this script with vim acting as vi, then you will likely see output like the following –

```
$ sh test.sh
Vim: Warning: Input is not from a terminal
$
```

After running the script, you should see the following added to the file **test.txt** –

```
$ cat test.txt
This file was created automatically from
a shell script
$
```

## Discard the output

Sometimes you will need to execute a command, but you don't want the output displayed on the screen. In such cases, you can discard the output by redirecting it to the file **/dev/null** –

```
$ command > /dev/null
```

Here **command** is the name of the command you want to execute. The file **/dev/null** is a special file that automatically discards all its input.

To discard both output of a command and its error output, use standard redirection to redirect **STDERR** to **STDOUT** –

```
$ command > /dev/null 2>&1
```

Here **2** represents **STDERR** and **1** represents **STDOUT**. You can display a message on to STDERR by redirecting STDOUT into STDERR as follows –

```
$ echo message 1>&2
```

## Redirection Commands

Following is a complete list of commands which you can use for redirection –

Sr.No.	Command & Description
1	<b>pgm &gt; file</b> Output of pgm is redirected to file
2	<b>pgm &lt; file</b> Program pgm reads its input from file
3	<b>pgm &gt;&gt; file</b> Output of pgm is appended to file
4	<b>n &gt; file</b> Output from stream with descriptor <b>n</b> redirected to file
5	<b>n &gt;&gt; file</b>

	Output from stream with descriptor <b>n</b> appended to file
6	<b>n &gt;&amp; m</b> Merges output from stream <b>n</b> with stream <b>m</b>
7	<b>n &lt;&amp; m</b> Merges input from stream <b>n</b> with stream <b>m</b>
8	<b>&lt;&lt; tag</b> Standard input comes from here through next tag at the start of line
9	<b> </b> Takes output from one program, or process, and sends it to another

Note that the file descriptor **0** is normally standard input (STDIN), **1** is standard output (STDOUT), and **2** is standard error output (STDERR).





**S.P. Mandali Pune  
Prin. K. P. Mangalvedhekar Institute of  
Management Career Development and  
Research.**

**156-B Railway Lines Solapur.  
Affiliated PAH Solapur University**

Name of Faculty . Mr Santosh Kulkarni

Subject Networking  
Programme:- BCA II Sem III

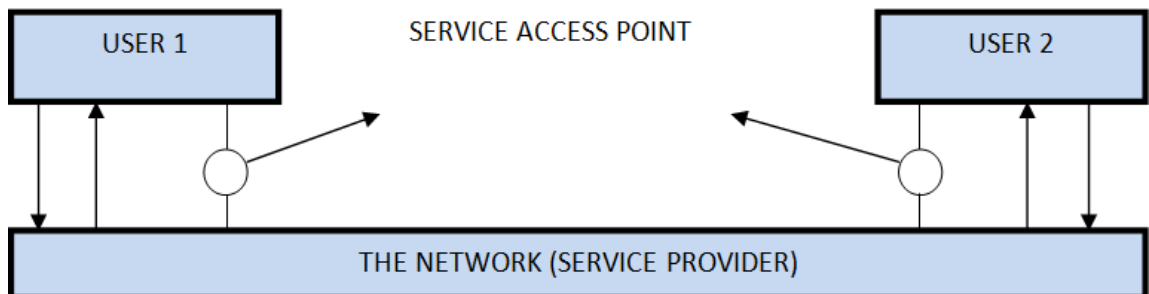
### **Network Models: Protocols & Standards,**

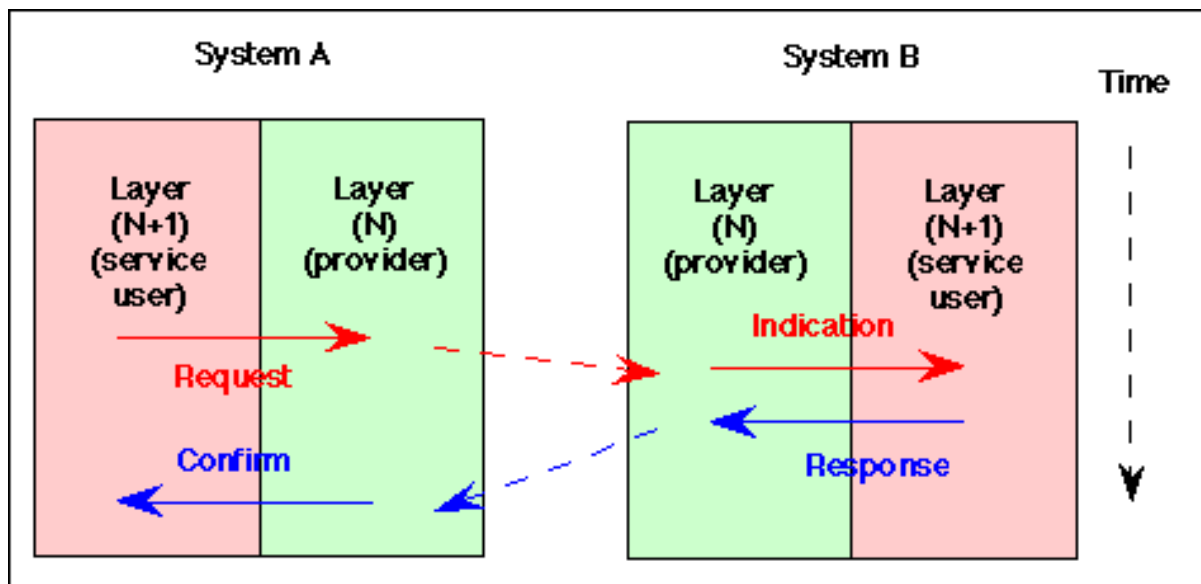
- **Network Protocols** are a set of guidelines governing the exchange of information in a simple, dependable and secure way. Network protocols are formal standards and policies comprised of rules, methodology, and configurations that define communication between two or more devices over a network. To effectively send and receive information, devices on the two sides of a communication exchange must follow protocols
- The modern computer network is designed around the concept of layered protocols of functions which meet the following goals
- 1) Provide a logical composition of a complex network in to smaller,more understandable parts
- 2) provide for standard interfaces between network functions for example standard interfaces between software program modules
- 3)Provide for symmetry in functions performed at each node in the network , Each layer performs the same functions as its counterpart in other nodes of the network
- 4) Provide a standard language to clarify communications between among network designers, vendors and users

### **Design Issues of Layers**

- Networks require the writing of hundreds of software programs. Such complex systems have bugs which is difficult to find and solve
- During the past several years as networks have grown in size and complexity the supporting communications software ,hardware have grown in size and function. Network maintenance has often experienced serious problems when changes were made that resulted in unpredictable consequences.
- Many networks have evolved without any standards by which to design them. The components within the network system sometimes have had poorly defined interfaces.
- The vendors approach to designing systems has been another major problem in networks. Each vendor had its own approach to designing networks. Developing the hardware and software for the networks and establishing how end user interfaces were integrated into the networks . So each vendor provided different and unique approach The basic idea of common standards is to develop a core of approaches among all vendors and to provide reliable network to all users
- Services Primitives

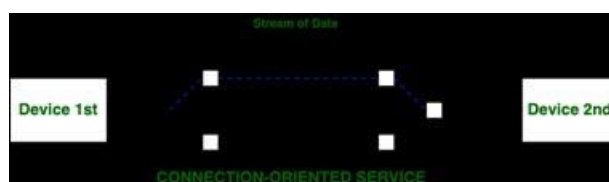
- In layered protocols a layer is service provider and may contain of several service functions. A function is a subsystem of a layer. Each subsystem may also be made up of entities. A entity is a specialized module of a layer or subsystem.
- The diagram shows four transactions called primitives are invoked to and from the layer through identifiers called service access points
- 1) Request :- Primitive by service user to invoke a function.
- 2) Indication Primitive by service provider to invoke a function or indicate a function has been invoked at a service access point
- 3) Response:- Primitive by service user to complete a function previously invoked by an indication at that SAP
- Confirm:- Primitive by service provider to complete a function previously invoked by a request at that SAP.
- The following diagram shows a user application invokes a service provider function by sending a request to the next lower layer . This service request is affirmed by the service provider returning a confirm. If the service is going to provide a function for another user B ,the service provider must send an indication to B is required to provide response.In a second diagram The service provider is in the middle of the diagram , with users A and B on each side. The request is sent to the service provider which sends user B an indication . User B provides a response which is transmited through the service provider as a confirm to A



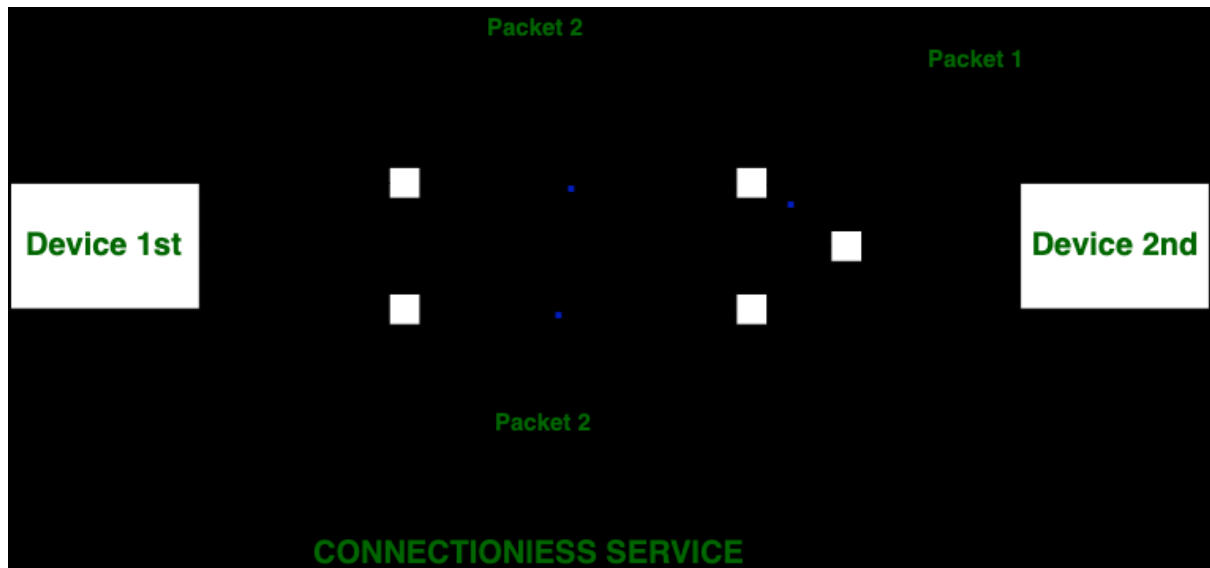


#### Connection oriented and connection less services

- Both Connection-oriented service and Connection-less service are used for the connection establishment between two or more than two devices. These type of services are offered by network layer. **Connection-oriented service** is related to the telephone system. It includes the connection establishment and connection termination. In connection-oriented service, Handshake method is used to establish the connection between sender and receiver.

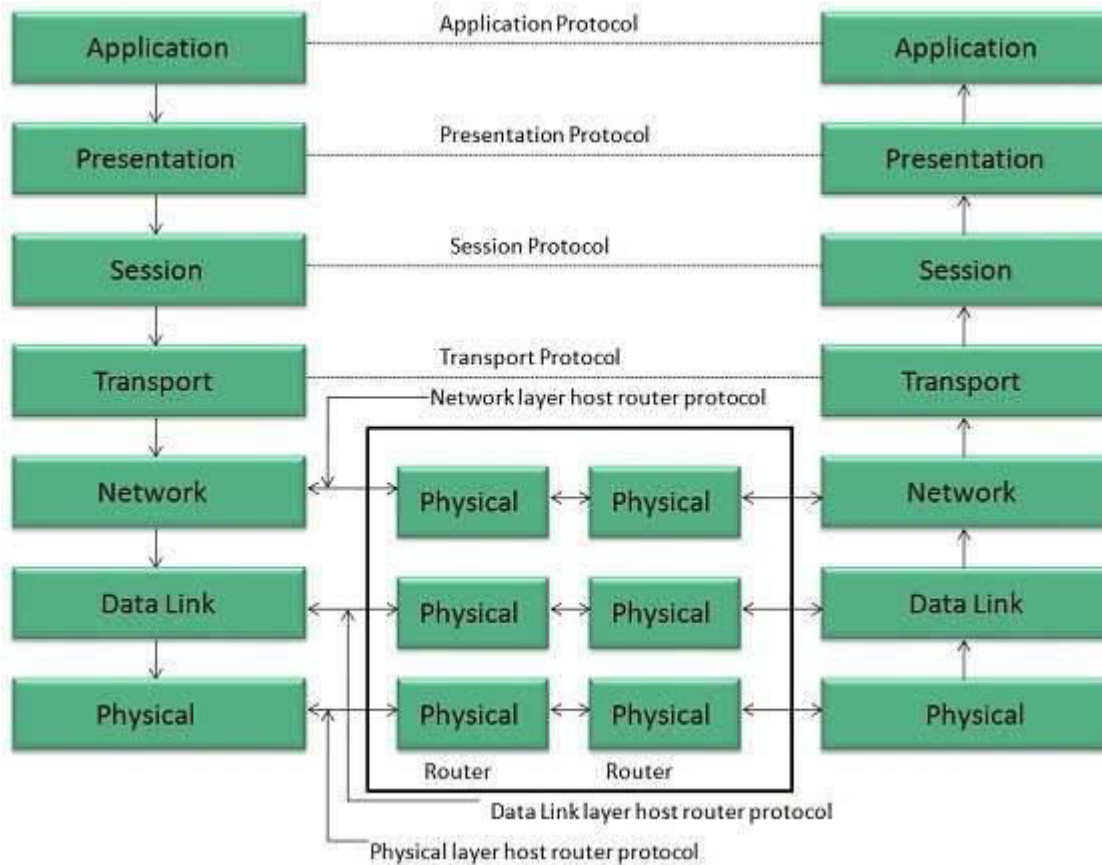


- 
- **Connection-less service** is related to the postal system. It does not include any connection establishment and connection termination. Connection-less Service does not give the guarantee of reliability. In this, Packets do not follow same path to reach destination.



#### ISO-OSI reference model

- Reference Model offers a means of standardization which is acceptable worldwide. Since people using the computer network are located over a wide physical range and their network devices might have heterogeneous architecture. In order to provide communication among heterogeneous devices, we need a standardized model i.e. a reference model, which would provide us way how these devices can communicate regardless their architecture.
- We have two reference models such as **OSI** model and **TCP/IP** reference model, however, the OSI model is a hypothetical one but the TCP/IP is absolutely practical model.
- OSI is acronym of Open System Interface. This model is developed by the International organization of Standardization (ISO) and therefore also referred as ISO-OSI Model.
- The OSI model consists of seven layers as shown in the following diagram. Each layer has a specific function, however each layer provide services to the layer above.



- Physical Layer
- The Physical layer is responsible for the following activities:
  - Activating, maintaining and deactivating the physical connection.
  - Defining voltages and data rates needed for transmission.
  - Converting digital bits into electrical signal.
  - Deciding whether the connection is simplex, half duplex or full duplex.
- Data Link Layer
- The data link layer performs the following functions:
  - Performs synchronization and error control for the information which is to be transmitted over the physical link.
  - Enables error detection, and adds error detection bits to the data which are to be transmitted.
- Network Layer

- Following are the functions of Network Layer:
- To route the signals through various channels to the other end.
- To act as the network controller by deciding which route data should take.

To divide the outgoing messages into packets and to assemble incoming packets into messages for higher levels

- Transport Layer
- The Transport layer performs the following functions:
  - It decides if the data transmission should take place on parallel paths or single path.
  - It performs multiplexing, splitting on the data.
  - It breaks the data groups into smaller units so that they are handled more efficiently by the network layer.
- Session Layer
- The Session layer performs the following functions:
  - Manages the messages and synchronizes conversations between two different applications.
  - It controls logging on and off, user identification, billing and session management.
- Presentation Layer
- The Presentation layer performs the following functions:
  - This layer makes it sure that the information is delivered in such a form that the receiving system will understand and use it.
- Application Layer
- The Application layer performs the following functions:
  - It provides different services such as manipulation of information in several ways, retransferring the files of information, distributing the results etc.
  - The functions such as LOGIN or password checking are also performed by the application layer.

TCP/IP reference model

- The **OSI Model** we just looked at is just a reference/logical model. It was designed to describe the functions of the communication system by dividing the communication procedure into smaller and simpler components. But when we talk about the TCP/IP model, it was designed and developed by Department of Defense (DoD) in 1960s and is based on standard protocols. It stands for Transmission Control Protocol/Internet Protocol.

The **TCP/IP model** is a concise version of the OSI model. It contains four layers, unlike seven layers in the OSI model. The layers are:

- Process/Application Layer
- Host-to-Host/Transport Layer
- Internet Layer
- Network Access/Link Layer

TCP/IP MODEL
Application Layer
Transport Layer
Internet Layer
Network Access Layer

OSI MODEL
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

- Network Access Layer –
- This layer corresponds to the combination of Data Link Layer and Physical Layer of the OSI model. It looks out for hardware addressing and the protocols present in this layer allows for the physical transmission of data.
- We just talked about ARP being a protocol of Internet layer, but there is a conflict about declaring it as a protocol of Internet Layer or Network access layer. It is described as residing in layer 3, being encapsulated by layer 2 protocols.
- 2. Internet Layer –
- This layer parallels the functions of OSI's Network layer. It defines the protocols which are responsible for logical transmission of data over the entire network. The main protocols residing at this layer are :



- IP – stands for Internet Protocol and it is responsible for delivering packets from the source host to the destination host by looking at the IP addresses in the packet headers. IP has 2 versions:
- IPv4 and IPv6. IPv4 is the one that most of the websites are using currently. But IPv6 is growing as the number of IPv4 addresses are limited in number when compared to the number of users.
- ICMP – stands for Internet Control Message Protocol. It is encapsulated within IP datagrams and is responsible for providing hosts with information about network problems.
- ARP – stands for Address Resolution Protocol. Its job is to find the hardware address of a host from a known IP address. ARP has several types: Reverse ARP, Proxy ARP, Gratuitous ARP and Inverse ARP.
- 3. Host-to-Host Layer –
- This layer is analogous to the transport layer of the OSI model. It is responsible for end-to-end communication and error-free delivery of data. It shields the upper-layer applications from the complexities of data. The two main protocols present in this layer are :
- Transmission Control Protocol (TCP) – It is known to provide reliable and error-free communication between end systems. It performs sequencing and segmentation of data. It also has acknowledgment feature and controls the flow of the data through flow control mechanism. It is a very effective protocol but has a lot of overhead due to such features. Increased overhead leads to increased cost.
- User Datagram Protocol (UDP) – On the other hand does not provide any such features. It is the go-to protocol if your application does not require reliable transport as it is very cost-effective. Unlike TCP, which is connection-oriented protocol, UDP is connectionless.
- 4. Application Layer –
- This layer performs the functions of top three layers of the OSI model: Application, Presentation and Session Layer. It is responsible for node-to-node communication and controls user-interface specifications. Some of the protocols present in this layer are: HTTP, HTTPS, FTP, TFTP, Telnet, SSH, SMTP, SNMP, NTP, DNS, DHCP, NFS, X Window, LPD. Have a look at Protocols in Application Layer for some information about these protocols. Protocols other than those present in the linked article are :
- HTTP and HTTPS – HTTP stands for Hypertext transfer protocol. It is used by the World Wide Web to manage communications between web browsers and servers. HTTPS stands for HTTP-Secure. It is a combination of HTTP with SSL(Secure Socket Layer). It is efficient in cases where the browser need to fill out forms, sign in, authenticate and carry out bank transactions.
- SSH – SSH stands for Secure Shell. It is a terminal emulations software similar to Telnet. The reason SSH is more preferred is because of its ability to maintain the encrypted connection. It sets up a secure session over a TCP/IP connection.

- NTP – NTP stands for Network Time Protocol. It is used to synchronize the clocks on our computer to one standard time source. It is very useful in situations like bank transactions. Assume the following situation without the presence of NTP. Suppose you carry out a transaction, where your computer reads the time at 2:30 PM while the server records it at 2:28 PM. The server can crash very badly if it's out of sync.

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5  
273**

**@gmail.com**



**S.P. Mandali Pune**  
**Prin. K. P. Mangalvedhekar Institute of Management**  
**Career Development and Research.**

**156-B Railway Lines Solapur**  
**Affiliated PAH Solapur University**

**Name of Faculty . Mr Santosh Kulkarni**

**Subject Cryptography**  
**Programme:- BCAIII Sem VI**

# Cryptogrphy

- IT is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.
- In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

# Techniques used For Cryptography:

- In today's age of computers cryptography is often associated with the process where an ordinary plain text is converted to cipher text which is the text made such that intended receiver of the text can only decode it and hence this process is known as encryption. The process of conversion of cipher text to plain text this is known as decryption.

# Features Of Cryptography are as follows:

- **Confidentiality:**  
Information can only be accessed by the person for whom it is intended and no other person except him can access it.
- **Integrity:**  
Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.
- **Non-repudiation:**  
The creator/sender of information cannot deny his or her intention to send information at later stage.
- **Authentication:**  
The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

- **Types Of Cryptography:**

In general there are three types Of cryptography:

- **Symmetric Key Cryptography:**

It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System(DES).



- **Asymmetric Key Cryptography:**

Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

- **Hash Functions:**

There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered.

Many operating systems use hash functions to encrypt passwords.

# Security goals

- In present day scenario security of the system is the sole priority of any organisation. The main aim of any organisation is to protect their data from attackers. In [cryptography](#), attacks are of two types such as [Passive attacks and Active attacks](#).
- Passive attacks are those that retrieve information from the system without affecting the system resources while active attacks are those that retrieve system information and make changes to the system resources and their operations.

- The Principles of Security can be classified as follows:
- **Confidentiality:**

The degree of confidentiality determines the secrecy of the information. The principle specifies that only the sender and receiver will be able to access the information shared between them. Confidentiality compromises if an unauthorized person is able to access a message. For example, let us consider sender A wants to share some confidential information with receiver B and the information gets intercepted by the attacker C. Now the confidential information is in the hands of an intruder C.

- **Authentication:**

Authentication is the mechanism to identify the user or system or the entity. It ensures the identity of the person trying to access the information. The authentication is mostly secured by using username and password. The authorized person whose identity is preregistered can prove his/her identity and can access the sensitive information.

- **Integrity:**

Integrity gives the assurance that the information received is exact and accurate. If the content of the message is changed after the sender sends it but before reaching the intended receiver, then it is said that the integrity of the message is lost.

- **Non-Repudiation:**

Non-repudiation is a mechanism that prevents the denial of the message content sent through a network. In some cases the sender sends the message and later denies it. But the non-repudiation does not allow the sender to refuse the receiver.

- **Access control:**

The principle of access control is determined by role management and rule management. Role management determines who should access the data while rule management determines up to what extent one can access the data. The information displayed is dependent on the person who is accessing it.

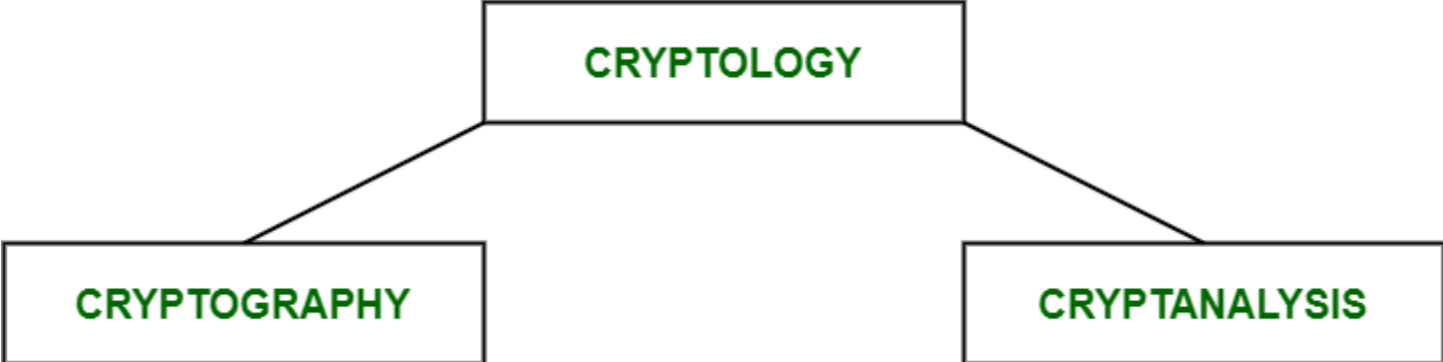


- **Availability:**

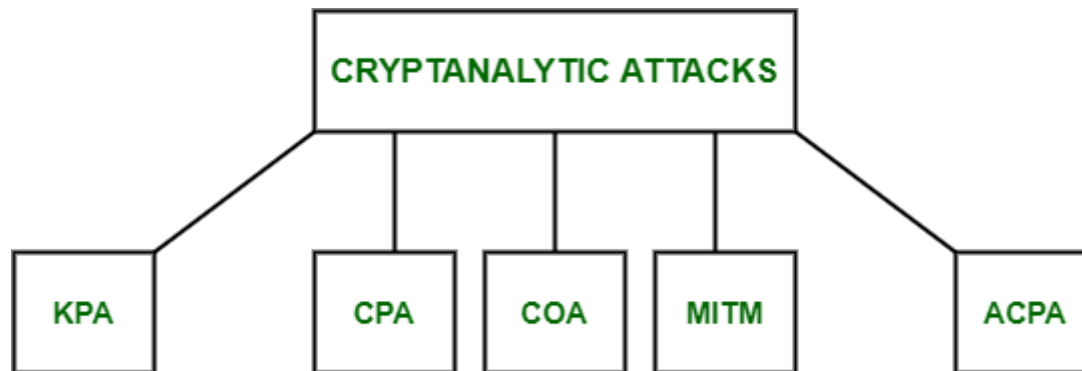
The principle of availability states that the resources will be available to authorize party at all times. Information will not be useful if it is not available to be accessed. Systems should have sufficient availability of information to satisfy the user request.

# Cryptographic Attacks,

- **Cryptology** has two parts namely, **Cryptography** which focuses on creating secret codes and **Cryptanalysis** which is the study of the cryptographic algorithm and the breaking of those secret codes. The person practicing Cryptanalysis is called a **Cryptanalyst**. It helps us to better understand the cryptosystems and also helps us improve the system by finding any weak point and thus work on the algorithm to create a more secure secret code. For example, a Cryptanalyst might try to decipher a ciphertext to derive the plaintext. It can help us to deduce the plaintext or the encryption key.



- To determine the weak points of a cryptographic system, it is important to attack the system. These attacks are called **Cryptanalytic attacks**. The attacks rely on the nature of the algorithm and also knowledge of the general characteristics of the plaintext, i.e., plaintext can be a regular document written in English or it can be a code written in Java. Therefore, the nature of the plaintext should be known before trying to use the attacks.

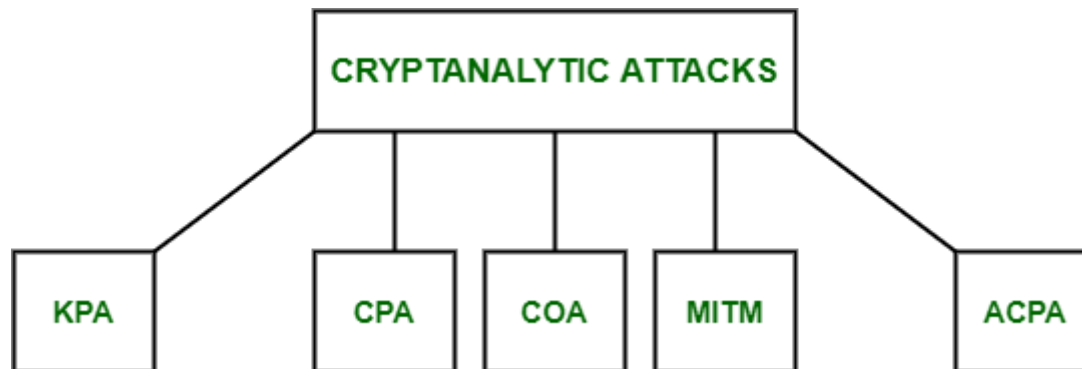


- Known-Plaintext Analysis (KPA) :
- In this type of attack, some plaintext-ciphertext pairs are already known. Attacker maps them in order to find the encryption key. This attack is easier to use as a lot of information is already available.
- Chosen-Plaintext Analysis (CPA) :
- In this type of attack, the attacker chooses random plaintexts and obtains the corresponding ciphertexts and tries to find the encryption key. Its very simple to implement like KPA but the success rate is quite low.

- Ciphertext-Only Analysis (COA) :
- In this type of attack, only some cipher-text is known and the attacker tries to find the corresponding encryption key and plaintext. Its the hardest to implement but is the most probable attack as only ciphertext is required.
- Man-In-The-Middle (MITM) attack :
- In this type of attack, attacker intercepts the message/key between two communicating parties through a secured channel.
- Adaptive Chosen-Plaintext Analysis (ACPA) :
- This attack is similar CPA. Here, the attacker requests the cipher texts of additional plaintexts after they have ciphertexts for some texts.

- To determine the weak points of a cryptographic system, it is important to attack the system. These attacks are called **Cryptanalytic attacks**. The attacks rely on the nature of the algorithm and also knowledge of the general characteristics of the plaintext, i.e., plaintext can be a regular document written in English or it can be a code written in Java. Therefore, the nature of the plaintext should be known before trying to use the attacks.

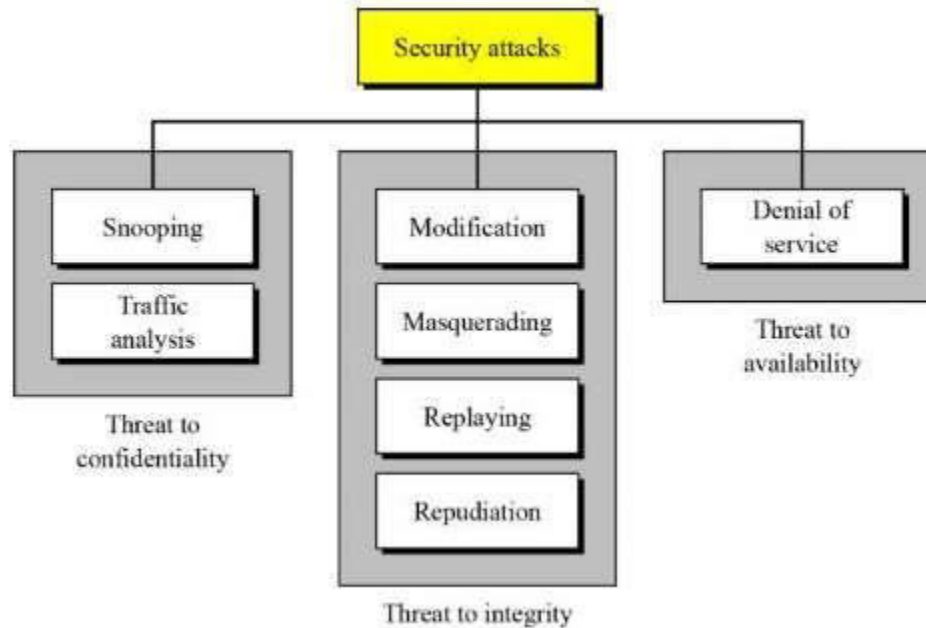




- **Known-Plaintext Analysis (KPA) :**  
In this type of attack, some plaintext-ciphertext pairs are already known. Attacker maps them in order to find the encryption key. This attack is easier to use as a lot of information is already available.
- **Chosen-Plaintext Analysis (CPA) :**  
In this type of attack, the attacker chooses random plaintexts and obtains the corresponding ciphertexts and tries to find the encryption key. Its very simple to implement like KPA but the success rate is quite low.

- **Ciphertext-Only Analysis (COA) :**  
In this type of attack, only some cipher-text is known and the attacker tries to find the corresponding encryption key and plaintext. Its the hardest to implement but is the most probable attack as only ciphertext is required.
- **Man-In-The-Middle (MITM) attack :**  
In this type of attack, attacker intercepts the message/key between two communicating parties through a secured channel.
- **Adaptive Chosen-Plaintext Analysis (ACPA) :**  
This attack is similar CPA. Here, the attacker requests the cipher texts of additional plaintexts after they have ciphertexts for some texts.

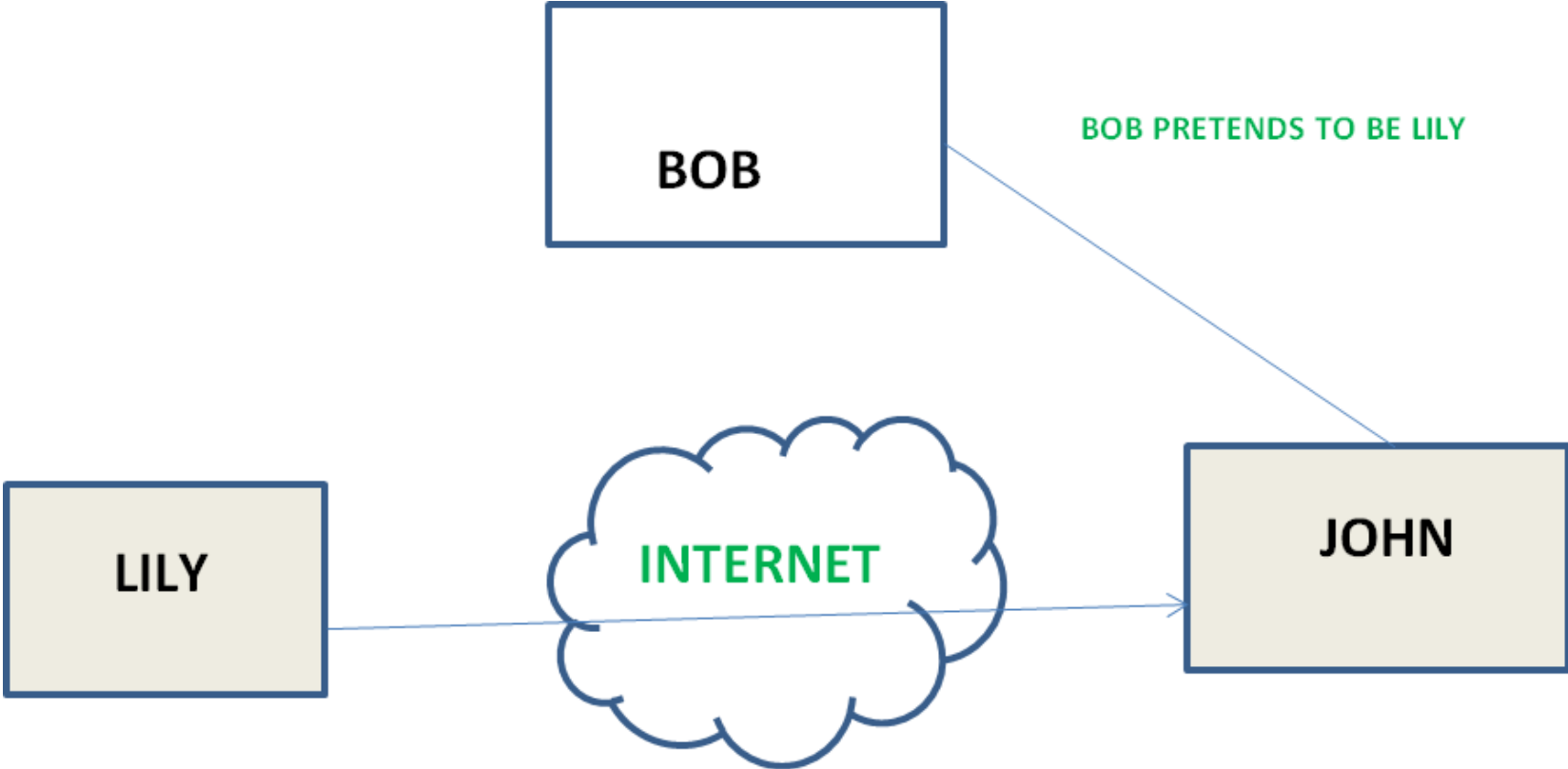
# Non Cryptanalytic Attacks



- **Masquerade –**

Masquerade attack takes place when one entity pretends to be different entity. A

Masquerade attack involves one of the other form of active attacks.

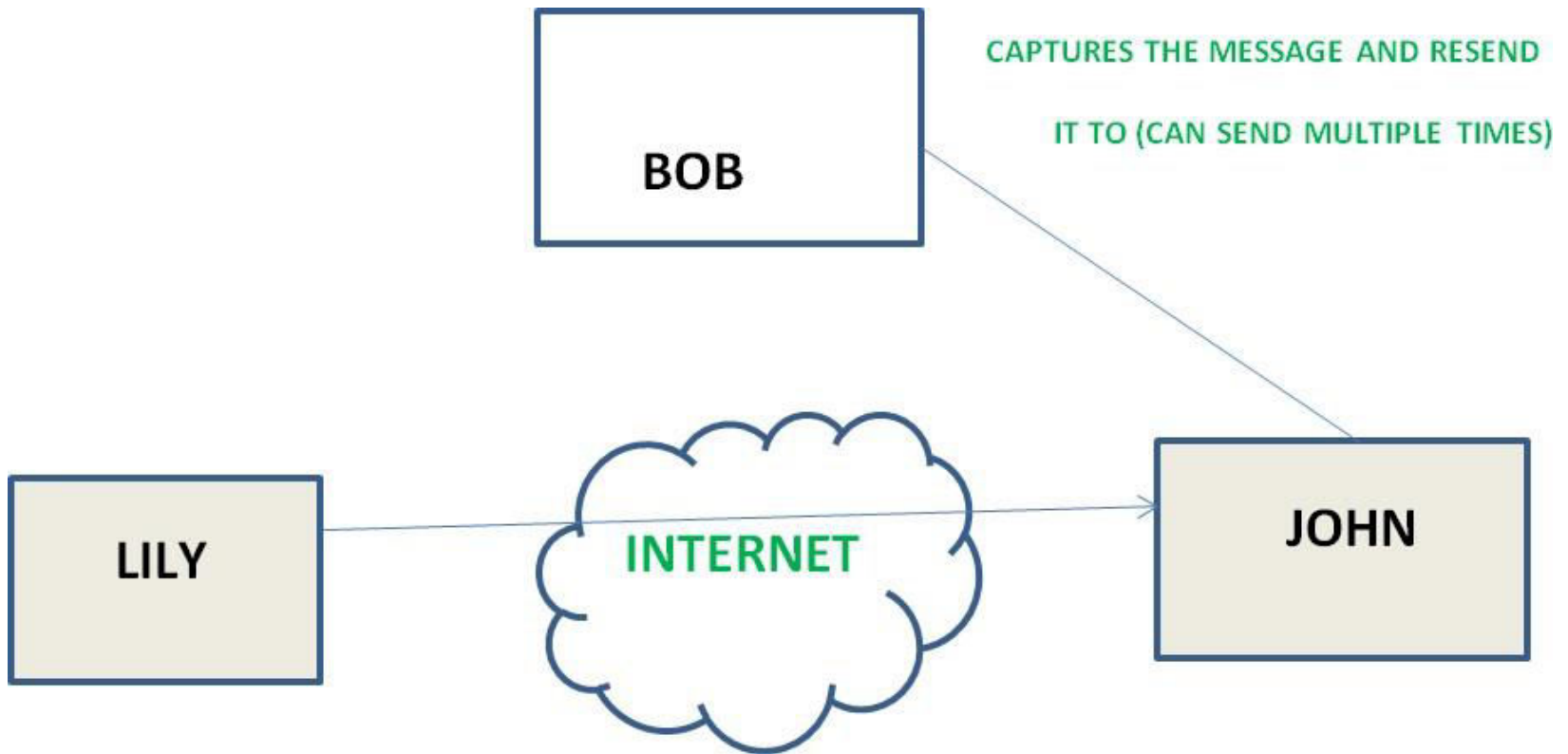


- **Repudiation –**

This attack is done by either sender or receiver. The sender or receiver can deny later that he/she has send or receive a message. For example, customer ask his Bank “To transfer an amount to someone” and later on the sender(customer) deny that he had made such a request. This is repudiation.

- **Replay –**

It involves the passive capture of a message and its subsequent the transmission to produce an authorized effect.



**BOB**

CAPTURES THE MESSAGE AND RESEND  
IT TO (CAN SEND MULTIPLE TIMES)

**LILY**

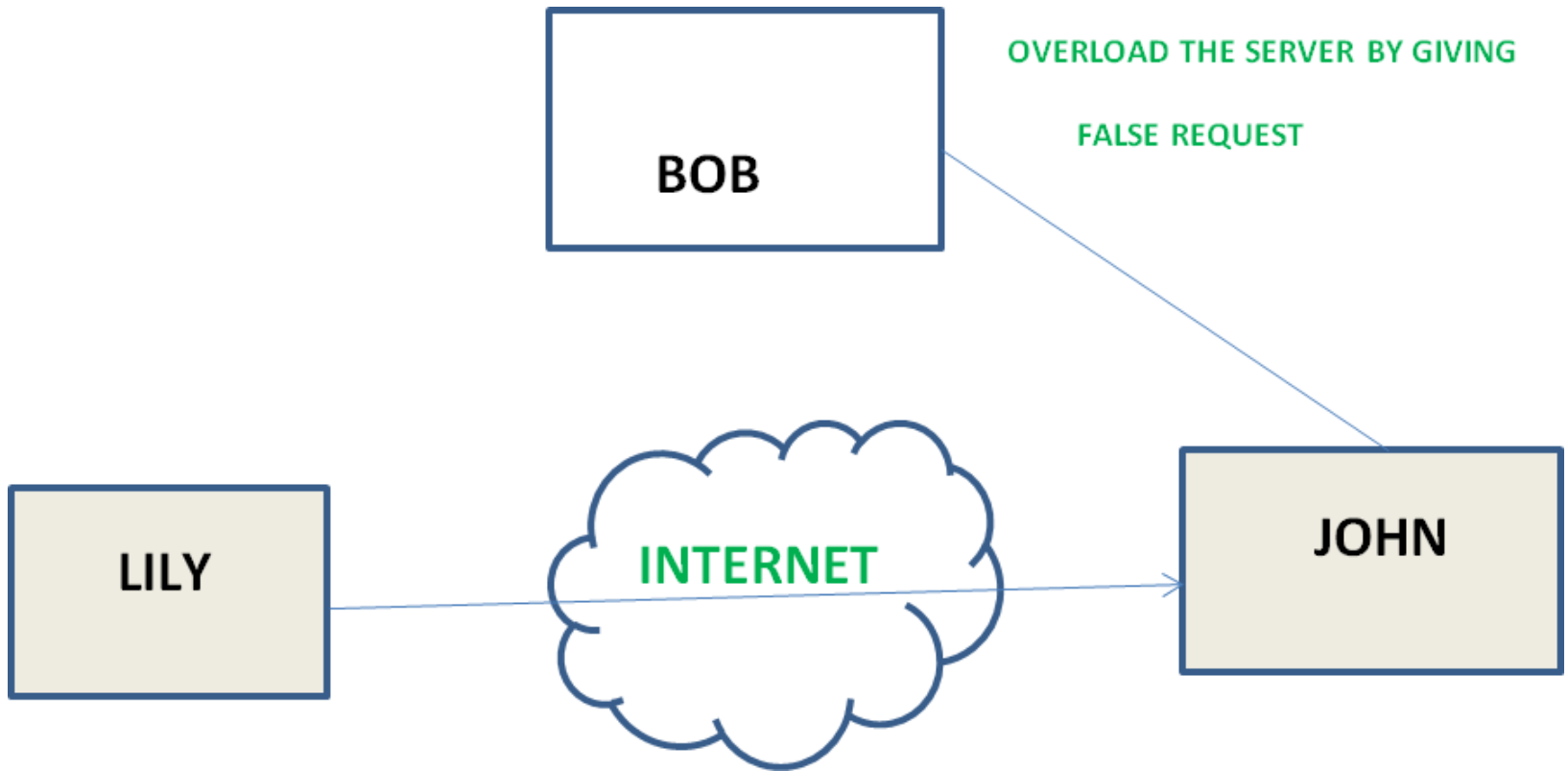
**INTERNET**

**JOHN**



- **Denial of Service –**

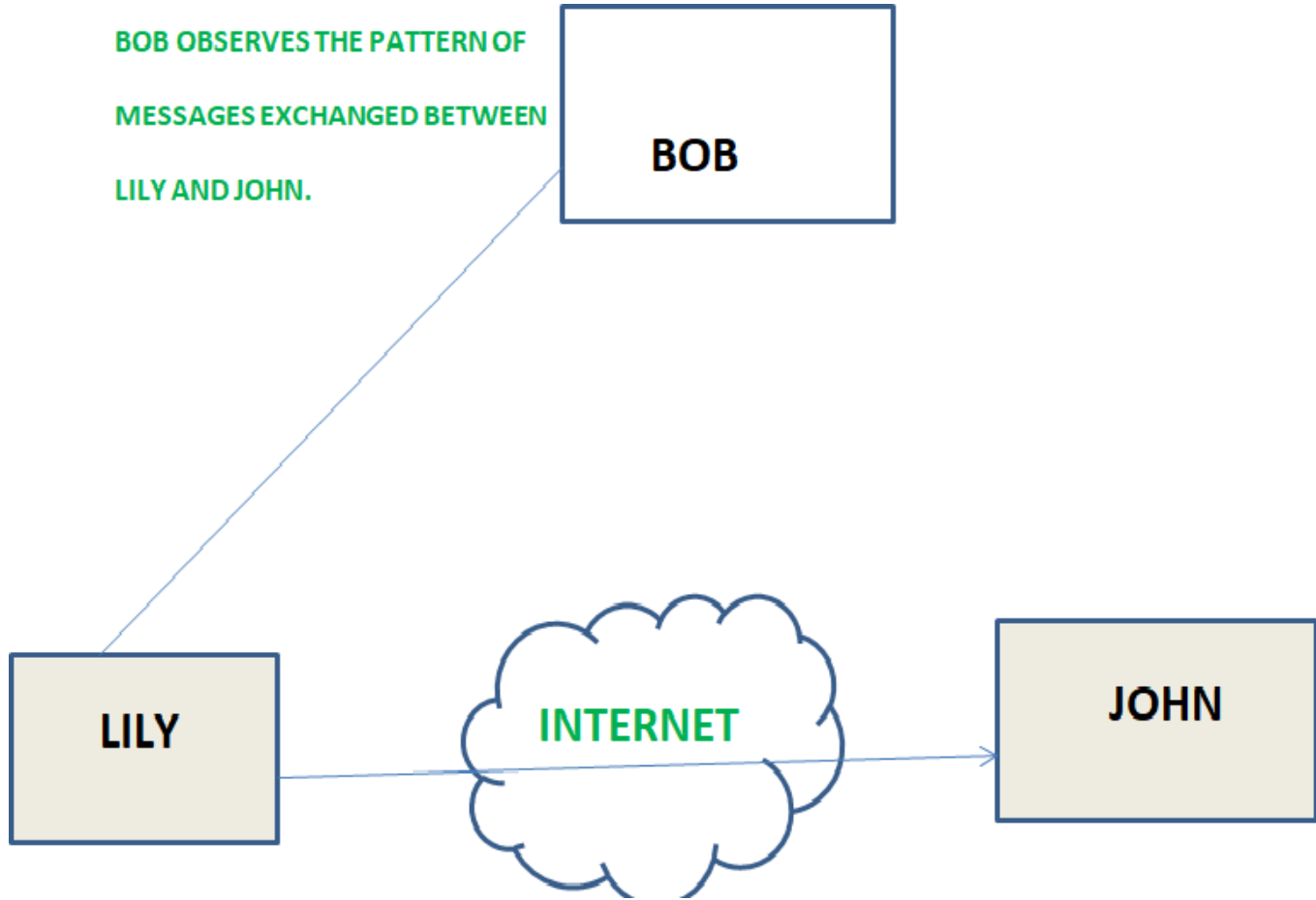
It prevents normal use of communication facilities. This attack may have a specific target. For example, an entity may suppress all messages directed to a particular destination. Another form of service denial is the disruption of an entire network either by disabling the network or by overloading it by messages so as to degrade performance.



- **Traffic analysis –**

Suppose that we had a way of masking (encryption) of information, so that the attacker even if captured the message could not extract any information from the message.

The opponent could determine the location and identity of communicating host and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.



**BOB OBSERVES THE PATTERN OF  
MESSAGES EXCHANGED BETWEEN  
LILY AND JOHN.**

**BOB**

**LILY**

**INTERNET**

**JOHN**

- Snooping
- Snooping refers to unauthorized access to or interception of data .for example a file transferred through the internet may contain confidential information. An unauthorized entity may intercept the transformation and use the contents for her own benefit.

# Services and mechanism

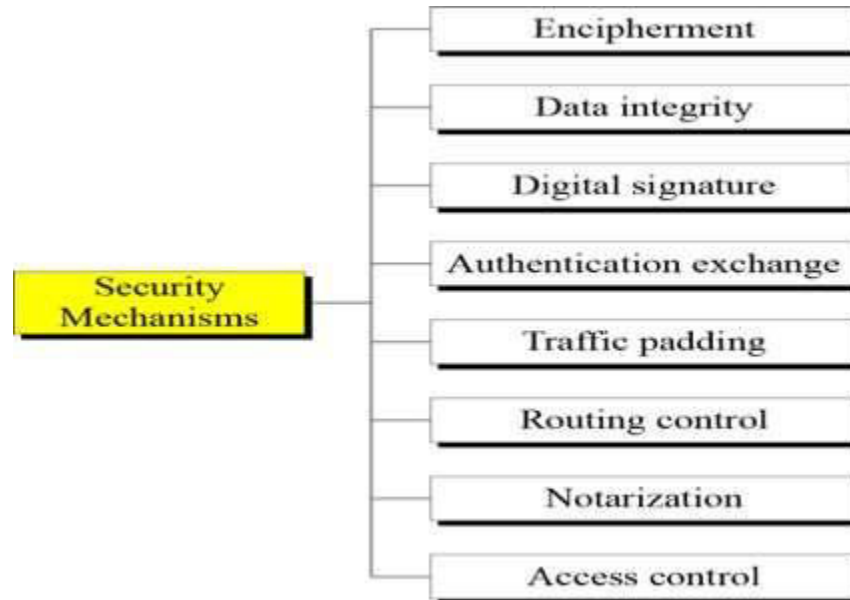
- ITU-T provides some security services and some mechanisms to implement those services.
- Security services and mechanisms are closely related because a mechanism or combination of mechanisms are used to provide a service



# Security Services

- **Authentication:** assures recipient that the **message is from the source** that it **claims to** be from.
- **Access Control:** controls who can have **access to resource** under what **condition**
- **Availability:** available to authorized entities for 24/7.
- **Confidentiality:** information is not made available to unauthorized individual
- **Integrity:** assurance that the message is unaltered
- **Non-Repudiation:** protection against denial of sending or receiving in the communication





<i>Security Service</i>	<i>Security Mechanism</i>
Data confidentiality	Encipherment and routing control
Data integrity	Encipherment, digital signature, data integrity
Authentication	Encipherment, digital signature, authentication exchanges
Nonrepudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

- **Confidentiality:**

The degree of confidentiality determines the secrecy of the information. The principle specifies that only the sender and receiver will be able to access the information shared between them. Confidentiality compromises if an unauthorized person is able to access a message. For example, let us consider sender A wants to share some confidential information with receiver B and the information gets intercepted by the attacker C. Now the confidential information is in the hands of an intruder C

- **Authentication:**

Authentication is the mechanism to identify the user or system or the entity. It ensures the identity of the person trying to access the information. The authentication is mostly secured by using username and password. The authorized person whose identity is preregistered can prove his/her identity and can access the sensitive information.

- **Integrity:**

Integrity gives the assurance that the information received is exact and accurate. If the content of the message is changed after the sender sends it but before reaching the intended receiver, then it is said that the integrity of the message is lost.

- **Non-Repudiation:**

Non-repudiation is a mechanism that prevents the denial of the message content sent through a network. In some cases the sender sends the message and later denies it. But the non-repudiation does not allow the sender to refuse the receiver.

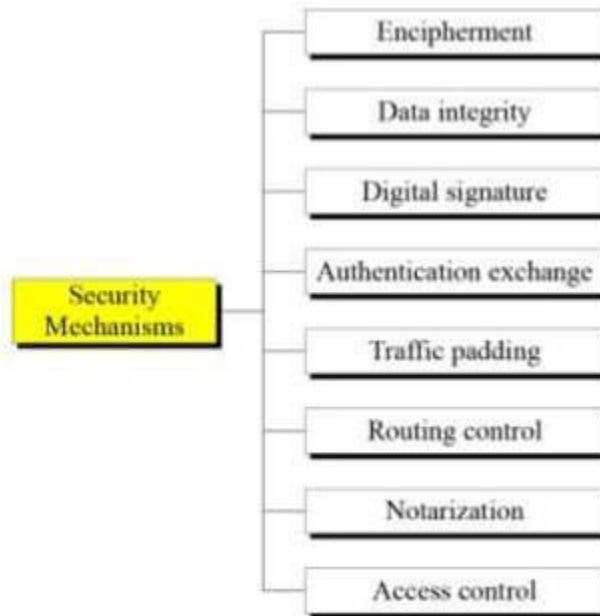
- **Access control:**

The principle of access control is determined by role management and rule management. Role management determines who should access the data while rule management determines up to what extent one can access the data. The information displayed is dependent on the person who is accessing it.

# Security Mechanisms

- is field in computer technology that deals with ensuring security of computer network infrastructure. As the network is very necessary for sharing of information whether it is at hardware level such as printer, scanner, or at software level. Therefore security mechanism can also be termed as is set of processes that deal with recovery from security attack. Various mechanisms are designed to recover from these specific attacks at various protocol layers.





- **Types of Security Mechanism are :**

- **Encipherment :**

This security mechanism deals with hiding and covering of data which helps data to become confidential. It is achieved by applying mathematical calculations or algorithms which reconstruct information into not readable form. It is achieved by two famous techniques named Cryptography and Encipherment. Level of data encryption is dependent on the algorithm used for encipherment.

- **Access Control :**

This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewall, or just by adding PIN to data.

- **Notarization :**

This security mechanism involves use of trusted third party in communication. It acts as mediator between sender and receiver so that if any chance of conflict is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

- **Data Integrity :**

This security mechanism is used by appending value to data to which is created by data itself. It is similar to sending packet of information known to both sending and receiving parties and checked before and after data is received. When this packet or data which is appended is checked and is the same while sending and receiving data integrity is maintained.

- **Authentication exchange :**

This security mechanism deals with identity to be known in communication. This is achieved at the TCP/IP layer where two-way handshaking mechanism is used to ensure data is sent or not

- **Digital Signature :**

This security mechanism is achieved by adding digital data that is not visible to eyes. It is form of electronic signature which is added by sender which is checked by receiver electronically. This mechanism is used to preserve data which is not more confidential but sender's identity is to be notified.

**Thank You !!!**

**Any Queries**

**9420779969/kulkarnisantosh5273**

**@gmail.com**

# Introduction of AngularJS

By Rapelli Shrushti

# Objective

To understand basics of angular, directives and expression

# Overview of AngularJS



# What is AngularJS?, Why AngularJS?

- ▶ Angular JS is an open source JavaScript framework to build web applications. It can be freely used, changed and shared by anyone.
- ▶ Angular Js is developed by Google.
- ▶ It is an excellent framework for building single phase applications and line of business applications.
- ▶ It was originally developed in 2009 by Misko Hevery and Adam Abrons.
- ▶ Its latest version is 1.2.21.
- ▶ Its data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

# Features of AngularJS

► The core features of AngularJS are as follows –

**Data-binding** – It is the automatic synchronization of data between model and view components.

**Scope** – These are objects that refer to the model. They act as a glue between controller and view.

**Controller** – These are JavaScript functions bound to a particular scope.

**Services** – AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.

**Filters** – These select a subset of items from an array and returns a new array.

**Directives** – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.

**Templates** – These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials.

**Routing** – It is concept of switching views.

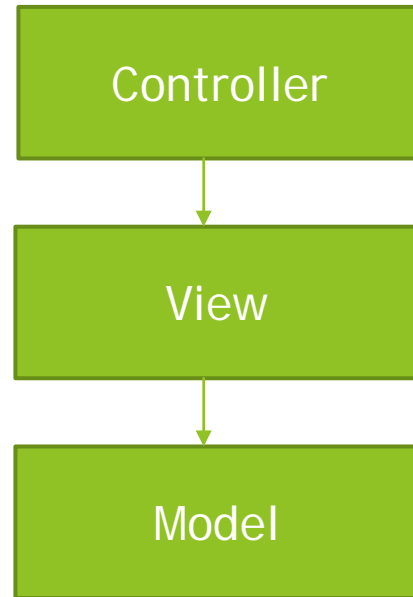
**Model View Whatever** – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.

**Deep Linking** – Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.

**Dependency Injection** – AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

# AngularJS architecture

- ▶ Angular.js follows the MVC architecture, the diagram of the MVC framework as shown below.



Angularjs Architecture Diagram

- ▶ The Controller represents the layer that has the business logic. User events trigger the functions which are stored inside your controller. The user events are part of the controller.
- ▶ Views are used to represent the presentation layer which is provided to the end users
- ▶ Models are used to represent your data. The data in your model can be as simple as just having primitive declarations. For example, if you are maintaining a student application, your data model could just have a student id and a name. Or it can also be complex by having a structured data model. If you are maintaining a car ownership application, you can have structures to define the vehicle itself in terms of its engine capacity, seating capacity, etc.

# AngularJS Advantages

- ▶ Since it's an open source framework, you can expect the number of errors or issues to be minimal.
- ▶ Two-way binding - Angular.js keeps the data and presentation layer in sync. Now you don't need to write additional JavaScript code to keep the data in your HTML code and your data later in sync. Angular.js will automatically do this for you. You just need to specify which control is bound to which part of your model.
- ▶ Routing - Angular can take care of routing which means moving from one view to another. This is the key fundamental of single page applications; wherein you can move to different functionalities in your web application based on user interaction but still stay on the same page.
- ▶ Angular supports testing, both Unit Testing, and Integration Testing.
- ▶ It extends HTML by providing its own elements called directives. At a high level, directives are markers on a DOM element (such as an attribute, element name, and comment or CSS class) that tell AngularJS's HTML compiler to attach a specified behavior to that DOM element. These directives help in extending the functionality of existing HTML elements to give more power to your web application.

# Setting up the Environment

- ▶ When you open the link <https://angularjs.org/>, you will see there are two options to download AngularJS library:
  1. View on GitHub - By clicking on this button, you are diverted to GitHub and get all the latest scripts.
  2. Download - By clicking on this button, a screen you get to see a dialog box
  3. Then We can go with the minimized, uncompressed, or zipped version.
  4. We extract it then we get CDN access.
  5. Copy angular.min.js and paste to your folder where ur programs saved.

# My first AngularJS app

```
<html>
<title>AngularJS First Application</title>
<body>
<h1>Sample Application</h1>
<div ng-app="">
  <p>Enter your Name: <input type="text" ng-model="name"></p>
  <p>Hello <span ng-bind="name"></span>!</p>
</div>
<script src="angular.min.js">
</script>
</body>
</html>
```

# Directives



# Introduction to Directives

- ▶ AngularJS facilitates you to extend HTML with new attributes. These attributes are called directives.
- ▶ There is a set of built-in directive in AngularJS which offers functionality to your applications. You can also define your own directives.
- ▶ AngularJS directives are used to extend HTML. They are special attributes starting with ng-prefix.
- ▶ Following are the most common directives:
  1. ng-app – This directive starts an AngularJS Application.
  2. ng-init – This directive initializes application data.
  3. ng-model – This directive defines the model that is variable to be used in AngularJS.
  4. ng-repeat – This directive repeats HTML elements for each item in a collection.

# Directive lifecycle

- ▶ The three phases of the life cycle of an AngularJS application happen each time a web page is loaded in the browser. The following sections describe these phases of an AngularJS application.

## The Bootstrap Phase

- ▶ The first phase of the AngularJS life cycle is the bootstrap phase, which occurs when the AngularJS JavaScript library is downloaded to the browser. AngularJS initializes its own necessary components and then initializes your module, which the ng-app directive points to. The module is loaded, and any dependencies are injected into your module and made available to code within the module.

## The Compilation Phase

- ▶ The second phase of the AngularJS life cycle is the HTML compilation stage. Initially when a web page is loaded, a static form of the DOM is loaded in the browser. During the compilation phase, the static DOM is replaced with a dynamic DOM that represents the AngularJS view.
- ▶ This phase involves two parts: traversing the static DOM and collecting all the directives and then linking the directives to the appropriate JavaScript functionality in the AngularJS built-in library or custom directive code. The directives are combined with a scope to produce the dynamic or live view.

## The Runtime Data Binding Phase

- ▶ The final phase of the AngularJS application is the runtime phase, which exists until the user reloads or navigates away from a web page. At that point, any changes in the scope are reflected in the view, and any changes in the view are directly updated in the scope, making the scope the single source of data for the view.
- ▶ AngularJS behaves differently from traditional methods of binding data. Traditional methods combine a template with data received from the engine and then manipulate the DOM each time the data changes. AngularJS compiles the DOM only once and then links the compiled template as necessary, making it much more efficient than traditional methods.

# Using AngularJS built-in directives

For in detail, visit <https://www.javatpoint.com/angularjs-directives>

## 1. Core Directives

### ng-app directive

- ▶ The ng-app directive starts an AngularJS Application. It defines the root element. It automatically initializes or bootstraps the application when the web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application. In the following example, we define a default AngularJS application using ng-app attribute of a <div> element.

```
<div ng-app = "">
```

```
...
```

```
</div>
```

### ng-init directive

The ng-init directive initializes an AngularJS Application data. It is used to assign values to the variables. In the following example, we initialize an array of countries. We use JSON syntax to define the array of countries.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'},  
  {locale:'en-GB',name:'United Kingdom'}, {locale:'en-FR',name:'France'}]">
```

```
...
```

```
</div>
```

### ► ng-model directive

The ng-model directive defines the model/variable to be used in AngularJS Application. In the following example, we define a model named name.

```
<div ng-app = "">  
  ...  
  <p>Enter your Name: <input type = "text" ng-model = "name"></p>  
</div>
```

### ► ng-repeat directive

The ng-repeat directive repeats HTML elements for each item in a collection. In the following example, we iterate over the array of countries.

```
<div ng-app = "">  
  ...  
  <p>List of Countries with locale:</p>  
  <ol>  
    <li ng-repeat = "country in countries">  
      {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}  
    </li>  
  </ol>  
</div>
```

► ng-model

```
<!DOCTYPE html>  
<html >  
<head>  
  <script src="angular.min.js"></script>  
</head>  
<body ng-app>  
  <input type="text" ng-model="name" />  
  <div>  
    Hello {{name}}  
  </div>  
</body>  
</html>
```

► ng-init

```
<!DOCTYPE html>
<html >
<head>
  <script src="angular.min.js"></script>
</head>
<body >
  <div ng-app ng-init="greet='Hello World!'; amount= 100; myArr = [100, 200]; person = {
firstName:'Steve', lastName : 'Jobs'}">
    {{amount}}    <br />
    {{myArr[1]}}  <br />
    {{person.firstName}}
  </div>
</body>
</html>
```

► ng-repeat

```
<!DOCTYPE html>
<html>
<head>
  <script src="angular.min.js"></script>
  <style>
    div {
      border: 1px solid green;
      width: 100%;
      height: 50px;
      display: block;
      margin-bottom: 10px;
      text-align:center;
      background-color:yellow;
    }
  </style>
</head>
<body ng-app="" ng-init="students=['Bill','Steve','Ram']">
  <ol>
    <li ng-repeat="name in students">
      {{name}}
    </li>
  </ol>
  <div ng-repeat="name in students">
    {{name}}
  </div>
</body>
</html>
```

# Conditional Directives

The ng-if directive removes the HTML element if the expression evaluates to false.

If the if statement evaluates to true, a copy of the Element is added in the DOM.

Syntax

```
<element ng-if="expression"></element>
```

Eg-

```
<!DOCTYPE html>
```

```
<html>
```

```
<script src="angular.min.js"></script>
```

```
<body ng-app="">
```

Keep HTML: 

```
<input type="checkbox" ng-model="myVar" ng-init="myVar = true">
```

```
<div ng-if="myVar">
```

```
<h1>Welcome</h1>
```

```
<p>Welcome to my home.</p>
```

```
<hr>
```

```
</div>
```

```
<p>The DIV element will be removed when the checkbox is not checked.</p>
```

```
<p>The DIV element will return, if you check the checkbox.</p>
```

```
</body>
```

```
</html>
```



# Style Directives

- ▶ The ng-style directive specifies the style attribute for the HTML element.
- ▶ The value of the ng-style attribute must be an object, or an expression returning an object.
- ▶ The object consists of CSS properties and values, in key value pairs.

## Syntax

```
<element ng-style="expression"></element>

<!DOCTYPE html>

<html>

<script src="angular.min.js"></script>

<body ng-app="myApp" ng-controller="myCtrl">

<h1 ng-style="myObj">Welcome</h1>

<script>

var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {

    $scope.myObj = {

        "color" : "white",

        "background-color" : "coral",

        "font-size" : "60px",

        "padding" : "50px"

    }

});

</script>

</body>

</html>
```

# Mouse and Keyboard Events Directives

## Mouse Events

Mouse events occur when the cursor moves over an element, in this order:

ng-mouseover

ng-mouseenter

ng-mousemove

ng-mouseleave

Or when a mouse button is clicked on an element, in this order:

ng-mousedown

ng-mouseup

ng-click

You can add mouse events on any HTML element

Eg

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<h1 ng-mousemove="count = count + 1">Mouse Over Me!</h1>
<h2>{{ count }}</h2>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>
</body>
</html>
```

In detail visit-[https://www.w3schools.com/angular/angular\\_events.asp](https://www.w3schools.com/angular/angular_events.asp)

## Keyevents

The ng-keypress directive tells AngularJS what to do when the keyboard is used on the specific HTML element.

The ng-keypress directive from AngularJS will not override the element's original onkeypress event, both will be executed.

The order of a key stroke is:

1. Keydown
2. Keypress
3. Keyup

## Syntax

```
<element ng-keypress="expression"></element>
```

Supported by `<input>`, `<select>`, `<textarea>`, and other editable elements.

```
<!DOCTYPE html>
```

```
<html>
```

```
<script src="angular.min.js"></script>
```

```
<body ng-app="">
```

```
<input ng-keypress="count = count + 1" ng-init="count=0" />
```

```
<h1>{{count}}</h1>
```

```
<p>This example will increase the value of the variable "count" every time a key is pressed in the input field.</p>
```

```
</body>
```

```
</html>
```

# Matching directives

The ng-switch directive lets you hide/show HTML elements depending on an expression.

Child elements with the ng-switch-when directive will be displayed if it gets a match, otherwise the element, and its children will be removed.

You can also define a default section, by using the ng-switch-default directive, to show a section if none of the other sections get a match.

## Syntax

```
<element ng-switch="expression">  
  <element ng-switch-when="value"></element>  
  <element ng-switch-when="value"></element>  
  <element ng-switch-when="value"></element>  
  <element ng-switch-default></element>  
</element>
```

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body ng-app="">
My favorite topic is:
<select ng-model="myVar">
  <option value="dogs">Dogs
  <option value="tuts">Tutorials
  <option value="cars">Cars
</select><hr>
<div ng-switch="myVar">
  <div ng-switch-when="dogs">
    <h1>Dogs</h1>
    <p>Welcome to a world of dogs.</p> </div>
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Learn from examples.</p> </div>
  <div ng-switch-when="cars">
    <h1>Cars</h1>
    <p>Read about cars.</p> </div>
  <div ng-switch-default>
    <h1>Switch</h1>
    <p>Select topic from the dropdown, to switch the content of this DIV.</p> </div>
</div>
<hr>
</body>
</html>
```

# Creating a custom directive

- ▶ In addition to all the built-in AngularJS directives, you can create your own directives.
- ▶ New directives are created by using the `.directive` function.
- ▶ To invoke the new directive, make an HTML element with the same tag name as the new directive.
- ▶ When naming a directive, you must use a camel case name, `w3TestDirective`, but when invoking it, you must use - separated name, `w3-test-directive`:



```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body ng-app="myApp">

<w3-test-directive></w3-test-directive>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
  return {
    template : "<h1>Made by a directive!</h1>"
  };
});
</script>

</body>
</html>
```

# Angular Expression

# All about Angular expressions

- ▶ AngularJS expressions can be written inside double braces: `{{ expression }}`.
- ▶ AngularJS expressions can also be written inside a directive: `ng-bind="expression"`.
- ▶ AngularJS will resolve the expression, and return the result exactly where the expression is written.
- ▶ AngularJS expressions are much like JavaScript expressions: They can contain literals, operators, and variables.

# How to use expressions

- ▶ Example {{ 5 + 5 }} or {{ firstName + " " + lastName }}

```
<!DOCTYPE html>
```

```
<html>
```

```
<script src="angular.min.js"></script>
```

```
<body>
```

```
<div ng-app>
```

```
<p>My first expression: {{ 5 + 5 }}</p>
```

```
</div>
```

```
</body>
```

```
</html>
```

# Number

```
<!DOCTYPE html>  
<html>  
<script src="angular.min.js"></script>  
<body>  
  
<div ng-app="" ng-init="quantity=1;cost=5">  
<p>Total in dollar: {{ quantity * cost }}</p>  
</div>  
  
</body>  
</html>
```

# String Expressions

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>

<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>The full name is: {{ firstName + " " + lastName }}</p>

</div>

</body>
</html>
```

# Object Binding and Expressions

Scope is a special JavaScript object that connects controller with views. Scope contains model data.

Data binding in AngularJS is the synchronization between the model and the view.

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
  <p ng-bind="firstname"></p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
  $scope.firstname = "John";
  $scope.lastname = "Doe";
});
</script>
<p>Use the ng-bind directive to bind the innerHTML of an element to a property in the data model.</p>
</body>
</html>
```

# Working with Arrays

- ▶ The ng-repeat directive repeats a set of HTML, a given number of times.
- ▶ The set of HTML will be repeated once per item in a collection.
- ▶ The collection must be an array or an object.
- ▶ Note: Each instance of the repetition is given its own scope, which consist of the current item.
- ▶ If you have an collection of objects, the ng-repeat directive is perfect for making a HTML table, displaying one table row for each object, and one table data for each object property. See example below.

- ▶ Syntax

```
<element ng-repeat="expression"></element>
```



```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>

<body ng-app="myApp" ng-controller="myCtrl">

<h1 ng-repeat="x in records">{{x}}</h1>

<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
  $scope.records = [
    "Alfreds Futterkiste",
    "Berglunds snabbköp",
    "Centro comercial Moctezuma",
    "Ernst Handel",
  ]
});
</script>

</body>
</html>
```

# Forgiving Behavior

AngularJS modifies the default behavior of some HTML elements.

Element	Description
a	AngularJS modifies the <a> element's default behaviors.
form	AngularJS modifies the <form> element's default behaviors.
input	AngularJS modifies the <input> element's default behaviors.
script	AngularJS modifies the <script> element's default behaviors.
select	AngularJS modifies the <select> element's default behaviors.
textarea	AngularJS modifies the <textarea> element's default behaviors.

# Angular expressions v/s Javascript expressions

- ▶ Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
- ▶ Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
- ▶ AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
- ▶ AngularJS expressions support filters, while JavaScript expressions do not.

What is PHP?

PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

Client-Side Scripting and Server-Side Scripting

Learn about Server-side and Client-side programming.

Introduction

Many websites out there use Server-side programming and Client-side programming for various functionalities. We can take the example of Scaler.com only. The static User Interface (UI) that can be seen by the user is developed using Client-side programming. Refer to the image below.

### **What is Website?**

Website is the collection of web pages, different multimedia content such as text, images, and videos which can be accessed by the URL which you can see in the address bar of the browser. For example: <https://www.geeksforgeeks.org>

### **How to access Websites?**

When we type a certain URL in a browser search bar, the browser requests the page from the Web server and the Web server returns the required web page and its content to the browser. Now, it differs how the server returns the information required in the case of static and dynamic websites.

### **Types of Website:**

- Static Website
- Dynamic Website

**Static Website:** In Static Websites, Web pages are returned by the server which are prebuilt source code files built using simple languages such as HTML, CSS, or JavaScript. There is no processing of content on the server (according to the user) in Static Websites. Web pages are returned by the server with no change therefore, static Websites are fast. There is no interaction with databases. Also, they are less costly as the host does not need to support server-side processing with different languages.

*Architecture of Static Website*

**Note:** Static does not mean that it will not respond to user actions, These Websites are called static because these cannot be manipulated on the server or interact with databases (which is the case in Dynamic Websites).

**Dynamic Website:** In Dynamic Websites, Web pages are returned by the server which are processed during runtime means they are not prebuilt web

pages but they are built during runtime according to the user's demand with the help of server-side scripting languages such as PHP, Node.js, ASP.NET and many more supported by the server. So, they are slower than static websites but updates and interaction with databases are possible.

Dynamic Websites are used over Static Websites as updates can be done very easily as compared to static websites (Where altering in every page is required) but in Dynamic Websites, it is possible to do a common change once and it will reflect in all the web pages.

### What Is Server-side Programming?

The execution of several programs on the back-end or specifically the servers, is said to be Server-side programming or Server-side scripting. Many functionalities are added in the web application for better user experience with the help of the programs that run on the backend. Hence, this is also called Back-end Programming. The main question is why do we need server-side or backend programming? Because when we deal with huge collections of data, we need servers that have databases to store such a huge amount of data. Then server-side programs are used to perform various operations on that data. In simple words, All the backend-code is maintained on the servers.

### Server-side Programming

In the above picture, you can see that the server is sending the response to the request made by the client. By this request and response cycle, client and server communicate with each other

PHP scripts can only be interpreted on a server that has PHP installed.

The client computers accessing the PHP scripts require a web browser only.

A PHP file contains PHP tags and ends with the extension “.php”.

In this tutorial, you will learn-

What is a Scripting Language?

Scripting VS Programming Language

What does PHP stand for?

PHP Syntax

### **Why use PHP?**

What is PHP used for & Market share

PHP vs ASP.NET VS JSP VS CFML

PHP File Extensions

PHP Hello world

### **What is a Scripting Language?**

A script is a set of programming instructions that is interpreted at runtime.

A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments.

The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.

Server side scripts are interpreted on the server while client side scripts are interpreted by the client application.

PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.

### **Programming Language Vs Scripting Language**

Programming language	Scripting language
----------------------	--------------------

Has all the features needed to develop complete applications.	Mostly used for routine tasks
---	-------------------------------

The code has to be compiled before it can be executed compiling	The code is usually executed without
--	--------------------------------------

Does not need to be embedded into other languages software environments.	Is usually embedded into other
---	--------------------------------

What does PHP stand for?

PHP means – Personal Home Page, but it now stands for the recursive backronym PHP: Hypertext Preprocessor.

PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks.

## PHP Syntax

What is PHP? Write your first PHP Program

A PHP file can also contain tags such as HTML and client side scripts such as JavaScript.

HTML is an added advantage when learning PHP Language. You can even learn PHP without knowing HTML but it's recommended you at least know the basics of HTML.

Database management systems DBMS for database powered applications.

For more advanced topics such as interactive applications and web services, you will need JavaScript and XML.

The flowchart diagram shown below illustrates the basic architecture of a PHP web application and how the server handles the requests.

What is PHP? Write your first PHP Program

Why use PHP?

You have obviously heard of a number of programming languages out there; you may be wondering why we would want to use PHP as our poison for the web programming. Below are some of the compelling reasons.

PHP is open source and free.

Short learning curve compared to other languages such as JSP, ASP etc.

Large community document

Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.

PHP is regular updated to keep abreast with the latest technology trends.

Other benefit that you get with PHP is that it's a server side scripting language; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.

PHP has in built support for working hand in hand with MySQL; this doesn't mean you can't use PHP with other database management systems. You can still use PHP with

Postgres

Oracle

MS SQL Server

ODBC etc.

PHP is cross platform; this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.

What is PHP used for & Market share

In terms of market share, there are over 20 million websites and application on the internet developed using PHP scripting language.

This may be attributed to the points raised above;

The diagram below shows some of the popular sites that use PHP

Web hosting    Supported by almost all hosting servers    Needs dedicated server    Fairly supported    Needs dedicated server

Open source

### **PHP File Extensions**

File extension and Tags In order for the server to identify our PHP files and scripts, we must save the file with the ".php" extension. Older PHP file extensions include

.phtml

.php3



.php4

.php5

.phps

PHP was designed to work with HTML, and as such, it can be embedded into the HTML code.

What is PHP? Write your first PHP Program

You can create PHP files without any html tags and that is called Pure PHP file .

The server interprets the PHP code and outputs the results as HTML code to the web browsers.

In order for the server to identify the PHP code from the HTML code, we must always enclose the PHP code in PHP tags.

A PHP tag starts with the less than symbol followed by the question mark and then the words "php".

PHP is a case sensitive language, "VAR" is not the same as "var".

The PHP tags themselves are not case-sensitive, but it is strongly recommended that we use lower case letter. The code below illustrates the above point.

```
<?php ... ?>
```

We will be referring to the PHP lines of code as statements. PHP statements end with a semi colon (;). If you only have one statement, you can omit the semi colon. If you have more than one statement, then you must end each line with a semi colon. For the sake of consistency, it is recommended that you always end your statement(s) with a semi colon. PHP scripts are executed on the server. The output is returned in form of HTML.

## PHP Hello world

The program shown below is a basic PHP application that outputs the words “Hello World!” When viewed in a web browser.

```
<?php  
echo "Hello world";  
?>
```

Output:

Hello world

## Summary

PHP stands for Hypertext pre-processor

PHP is a server side scripting language. This means that it is executed on the server. The client applications do not need to have PHP installed.

PHP files are saved with the “.php” file extension, and the PHP development code is enclosed in tags.

PHP is open source and cross platform

You Might Like:

[PHP Data Types, Variables, Constant, Operators Tutorial](#)

[PHP Date\(\) & Time Function: How to Get Current Timestamp?](#)

[PHP Function: Built in, String, Numeric with Examples](#)

[PHP preg\\_match\(\): Regular Expressions \(Regex\)](#)

[PHP Loop: For, ForEach, While, Do While \[Example\]](#)

What is XAMPP?

XAMPP is an open-source, cross-platform web server that consists of a web server, MySQL database engine, and PHP and Perl programming packages. It is compiled and maintained by Apache. It allows users to create WordPress websites online using a local web server on their computer. It supports Windows, Linux, and Mac.

It is compiled and maintained by apache. The acronym XAMPP stands for;

X – [cross platform operating systems] meaning it can run on any OS Mac OS X , Windows , Linux etc.

A – Apache – this is the web server software.

M – MySQL – Database.

P – PHP

P – Perl – scripting language

Why use XAMPP?

XAMPP provides an easy-to-use control panel to manage Apache, MySQL, and other programs without using commands. To use PHP, we need to install Apache and MySQL. It's not easy to install Apache and configure it as it needs to be set up and integrated with PHP and Perl, among other things. XAMPP deals with all the complexity to set up and integrate Apache with PHP and Perl.

Unlike Java that runs with the Java SDK only, PHP requires a web-server to work.

1) This section lists the installed services, modules and the process IDs PID(s). A green tick means the module has been installed as a service. The red mark means it has not been installed as a service. To install a service, click on the red mark. If the button shows a green tick and you click on it, the control panel will ask you if you want to uninstall the system.

2) This section shows Port(s) associated with the modules. The actions section is for;

starting and stopping modules

Open the administrative windows for Apache and MySQL

Open configuration files for Apache, MySQL etc. to make changes

View log files for the modules

3) This section contains useful utilities such as Netsat, windows services short cuts etc.

4) This section displays status information on the modules. The control panel can be used to;

Install and uninstall services such as Apache, MySQL etc. that are installed via XAMPP

Start and stop services.

Open configure files etc.

Configure XAMPP

Let's now look at the basic configurations required before we start using our XAMPP installation for developing PHP powered web sites. Type the URL <http://localhost/xampp/> in your favorite browser. For this tutorial, we will be using Firefox as our web browser.

Download and Install XAMPP & Netbeans

If you are able to see the above screen then you have installed XAMPP successfully. The panel on the left hand side contains links to useful information such as;

The version of PHP installed

Security settings of XAMPP

Access to utilities such as phpMyAdmin etc.

The PHP version shipped with XAMPP 1.8.0 is PHP 5.4.4

What is the best PHP IDE?

A PHP IDE is a program that allows you to easily write PHP codes. PHP IDEs are often equipped with syntax highlighting features and in some cases autocomplete features too. This means that if you write a PHP keyword that is known by the PHP interpreter, then the keyword will be highlighted a different color from the one used for regular statements. The autocomplete features automatically pops up known PHP keywords as you type them. Notepad can also be used to write and editor PHP codes. The disadvantage of using an editor such as Notepad is that debugging the scripts becomes difficult because it is not easy to spot errors such as misspelt keywords, unclosed braces etc. an IDE will highlight the

statements with errors so it's easy for you to spot them. The table shown below shows 5 popular PHP editors

Editor	License	Cross Platform	Brief description
--------	---------	----------------	-------------------

Netbeans IDE	Open Source	Yes	
--------------	-------------	-----	--

Dedicated PHP coding environment with syntax highlighting and code completion for keywords and other known information.

Supports integration with PHP MVC frameworks i.e. Zend,

Code History that shows the changes made to a file

SFTP,FTP and SVN via plugins.

Dreamweaver	Commercial	Yes	
-------------	------------	-----	--

Supports HTML and PHP.

Syntax highlighting, code folding and completion for keywords and other known information.

Supports SFTP and FTP.

Zend studio	Commercial	Yes	
-------------	------------	-----	--

Integrated with Zend Server and Zend PHP MVC framework, PHPUnit, phpDocumentor etc.

Has syntax highlighting, code folding,

Support for Web services etc.

PHP Eclipse	Open Source	Yes	
-------------	-------------	-----	--

Code formatter

Supports SVN, SHH/FTP

Notepad ++	Freeware	Windows only	
------------	----------	--------------	--

Syntax highlighting

Supports SFTP and FTP via plugins.

Netbeans IDE PHP editor As briefly highlighted in the above table, Netbeans IDE has powerful features that enhance the productive of PHP coders. The IDE can be freely downloaded from the <https://netbeans.org/downloads/index.html>

Syntax highlighting and auto-complete features enhances your productivity

It has native support for database systems like MySQL. You don't need to use two programs to code and develop your database.

The IDE can be used in a collaborative environment. This comes in handy when you have to work with other developers as a team.

The IDE has support for other languages such as;

Java SE

Java EE

C

C++

In PHP, a variable is declared using a **\$ sign** followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

1. `$variablename=value;`

Rules for declaring PHP variable:

- A variable must start with a dollar (\$) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, \_).
- A variable name must start with a letter or underscore (\_) character.
- A PHP variable name cannot contain spaces.

- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so \$name and \$NAME both are treated as different variable.

## PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

*File: variable1.php*

```
1.     <?php
2.     $str="hello string";
3.     $x=200;
4.     $y=44.6;
5.     echo "string is: $str <br/>";
6.     echo "integer is: $x <br/>";
7.     echo "float is: $y <br/>";
8.     ?>
```

### Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

## PHP Variable: Sum of two variables

*File: variable2.php*

```
1.     <?php
2.     $x=5;
3.     $y=6;
4.     $z=$x+$y;
5.     echo $z;
6.     ?>
```

### Output:

## PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
1.      <?php
2.      $color="red";
3.      echo "My car is " . $color . "<br>";
4.      echo "My house is " . $COLOR . "<br>";
5.      echo "My boat is " . $coLOR . "<br>";
6.      ?>
```

### Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

## PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
1.      <?php
2.      $a="hello";//letter (valid)
3.      $_b="hello";//underscore (valid)
4.
5.      echo "$a <br/> $_b";
6.      ?>
```

### Output:

```
hello
```



```
hello
```

File: *variableinvalid.php*

1. `<?php`
2. `$4c="hello";//number (invalid)`
3. `*$d="hello";//special symbol (invalid)`
- 4.
5. `echo "$4c <br/> $*d";`
6. `?>`

### Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VARIABLE) or '$' in C:\wamp\www\variableinvalid.php on line 2
```

## PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

---

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

## Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

File: local\_variable1.php

```
1.     <?php
2.         function local_var()
3.         {
4.             $num = 45; //local variable
5.             echo "Local variable declared inside the function is: ". $num;
6.         }
7.         local_var();
8.     ?>
```

### Output:

```
Local variable declared inside the function is: 45
```

File: local\_variable2.php

```
1.     <?php
2.         function mytest()
3.         {
4.             $lang = "PHP";
5.             echo "Web development language: " . $lang;
6.         }
7.         mytest();
8.         //using $lang (local variable) outside the function will generate an error
9.         echo $lang;
10.    ?>
```

### Output:

```
Web development language: PHP
Notice: Undefined variable: lang in D:\xampp\htdocs\program\p3.php on line 28
```

## Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Let's understand the global variables with the help of an example:

## Example:

File: `global_variable1.php`

```
1.     <?php
2.         $name = "Sanaya Sharma";    //Global Variable
3.         function global_var()
4.         {
5.             global $name;
6.             echo "Variable inside the function: ". $name;
7.             echo "</br>";
8.         }
9.         global_var();
10.        echo "Variable outside the function: ". $name;
11.    ?>
```

### Output:

```
Variable inside the function: Sanaya Sharma
Variable outside the function: Sanaya Sharma
```

## PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

```
1.     $num=10+20;//+ is the operator and 10,20 are operands
```

In the above example, + is the binary + operator, 10 and 20 are operands and \$num is variable.

PHP Operators can be categorized in following forms:

- [Arithmetic Operators](#)
- [Assignment Operators](#)
- [Bitwise Operators](#)
- [Comparison Operators](#)

- [Incrementing/Decrementing Operators](#)
- [Logical Operators](#)
- [String Operators](#)
- [Array Operators](#)
- [Type Operators](#)
- [Execution Operators](#)
- [Error Control Operators](#)

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

- **Unary Operators:** works on single operands such as ++, -- etc.
- **Binary Operators:** works on two operands such as binary +, -, \*, / etc.
- **Ternary Operators:** works on three operands such as "?:".

---

## Arithmetic Operators

The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	\$a + \$b	Sum of operands
-	Subtraction	\$a - \$b	Difference of operands
*	Multiplication	\$a * \$b	Product of operands
/	Division	\$a / \$b	Quotient of

			operands
%	Modulus	\$a % \$b	Remainder of operands
**	Exponentiation	\$a ** \$b	\$a raised to the power \$b

The exponentiation (\*\*) operator has been introduced in PHP 5.6.

## Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	\$a = \$b	The value of right operand is assigned to the left operand.
+=	Add then Assign	\$a += \$b	Addition same as \$a = \$a + \$b
-=	Subtract then Assign	\$a -= \$b	Subtraction same as \$a = \$a - \$b
*=	Multiply then Assign	\$a *= \$b	Multiplication same as \$a = \$a * \$b

/=	Divide then Assign (quotient)	$\$a /= \$b$	Find quotient same as $\$a = \$a / \$b$
%=	Divide then Assign (remainder)	$\$a \% = \$b$	Find remainder same as $\$a = \$a \% \$b$

## Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$\$a \& \$b$	Bits that are 1 in both $\$a$ and $\$b$ are set to 1, otherwise 0.
	Or (Inclusive or)	$\$a   \$b$	Bits that are 1 in either $\$a$ or $\$b$ are set to 1
^	Xor (Exclusive or)	$\$a \wedge \$b$	Bits that are 1 in either $\$a$ or $\$b$ are set to 0.
~	Not	$\sim \$a$	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$\$a \ll \$b$	Left shift the bits of

			operand \$a \$b steps
>>	Shift right	\$a >> \$b	Right shift the bits of \$a operand by \$b number of places

## Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	\$a == \$b	Return TRUE if \$a is equal to \$b
===	Identical	\$a === \$b	Return TRUE if \$a is equal to \$b, and they are of same data type
!==	Not identical	\$a !== \$b	Return TRUE if \$a is not equal to \$b, and they are not of same data type
!=	Not equal	\$a != \$b	Return TRUE if \$a is not equal to \$b
<>	Not equal	\$a <> \$b	Return TRUE if \$a is not equal to \$b

<	Less than	$\$a < \$b$	Return TRUE if \$a is less than \$b
>	Greater than	$\$a > \$b$	Return TRUE if \$a is greater than \$b
<=	Less than or equal to	$\$a \leq \$b$	Return TRUE if \$a is less than or equal to \$b
>=	Greater than or equal to	$\$a \geq \$b$	Return TRUE if \$a is greater than or equal to \$b
<=>	Spaceship	$\$a \lt;=> \$b$	Return -1 if \$a is less than \$b Return 0 if \$a is equal to \$b Return 1 if \$a is greater than \$b

## Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++\$a	Increment the value of \$a by one, then return \$a



		\$a++	Return \$a, then increment the value of \$a by one
--	decrement	--\$a	Decrement the value of \$a by one, then return \$a
		\$a--	Return \$a, then decrement the value of \$a by one

## Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	\$a and \$b	Return TRUE if both \$a and \$b are true
Or	Or	\$a or \$b	Return TRUE if either \$a or \$b is true
xor	Xor	\$a xor \$b	Return TRUE if either \$a or \$b is true but not both
!	Not	! \$a	Return TRUE if \$a is not true

&&	And	\$a && \$b	Return TRUE if either \$a and \$b are true
	Or	\$a    \$b	Return TRUE if either \$a or \$b is true

## String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	\$a . \$b	Concatenate both \$a and \$b
.=	Concatenation and Assignment	\$a .= \$b	First concatenate \$a and \$b, then assign the concatenated string to \$a, e.g. \$a = \$a . \$b

## Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
----------	------	---------	-------------

+	Union	<code>\$a + \$y</code>	Union of \$a and \$b
<code>==</code>	Equality	<code>\$a == \$b</code>	Return TRUE if \$a and \$b have same key/value pair
<code>!=</code>	Inequality	<code>\$a != \$b</code>	Return TRUE if \$a is not equal to \$b
<code>===</code>	Identity	<code>\$a === \$b</code>	Return TRUE if \$a and \$b have same key/value pair of same type in same order
<code>!==</code>	Non-Identity	<code>\$a !== \$b</code>	Return TRUE if \$a is not identical to \$b
<code>&lt;&gt;</code>	Inequality	<code>\$a &lt;&gt; \$b</code>	Return TRUE if \$a is not equal to \$b

## PHP If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

- [if](#)
- [if-else](#)
- [if-else-if](#)
- [nested if](#)

## PHP If Statement

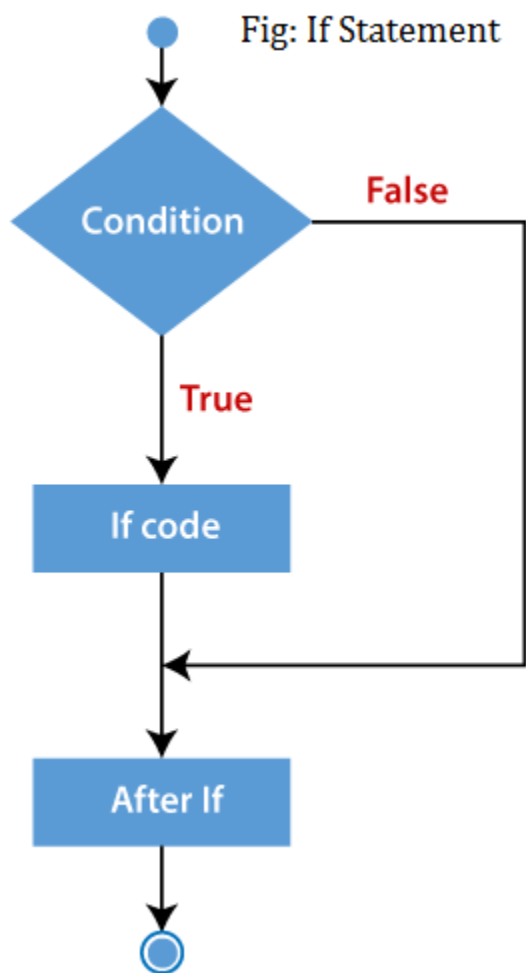
PHP if statement allows conditional execution of code. It is executed if condition is true.

If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

### Syntax

1. `if(condition){`
2. `//code to be executed`
3. `}`

### Flowchart



### Example

1. `<?php`
2. `$num=12;`

```
3.     if($num<100){
4.     echo "$num is less than 100";
5.     }
6.     ?>
```

### Output:

```
12 is less than 100
```

## PHP If-else Statement

PHP if-else statement is executed whether condition is true or false.

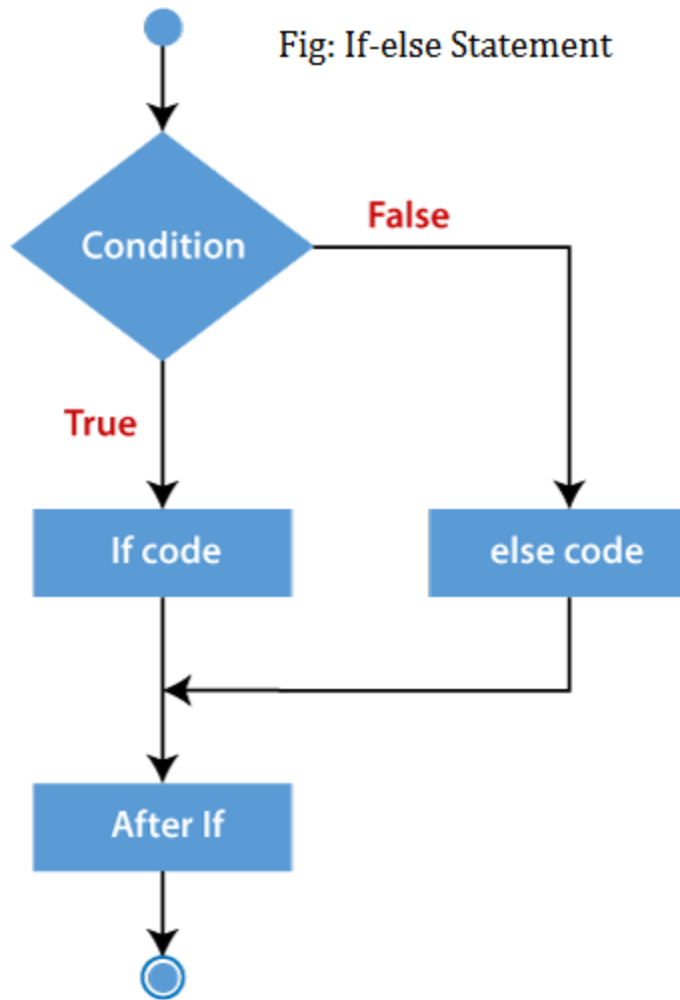
If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

### Syntax

```
1.     if(condition){
2.     //code to be executed if true
3.     }else{
4.     //code to be executed if false
5.     }
```

### Flowchart

Fig: If-else Statement



### Example

1. `<?php`
2. `$num=12;`
3. `if($num%2==0){`
4. `echo "$num is even number";`
5. `}else{`
6. `echo "$num is odd number";`
7. `}`
8. `?>`

### Output:

```
12 is even number
```

## PHP If-else-if Statement

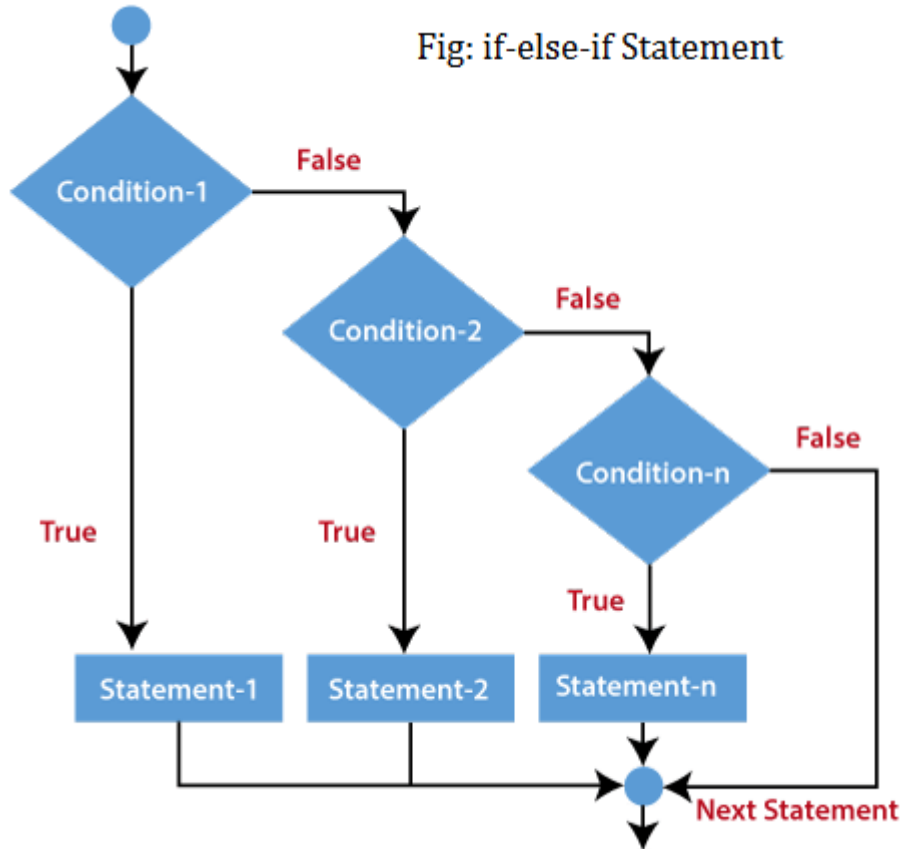
The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

### Syntax

```
1.      if (condition1){
2.      //code to be executed if condition1 is true
3.      } elseif (condition2){
4.      //code to be executed if condition2 is true
5.      } elseif (condition3){
6.      //code to be executed if condition3 is true
7.      ....
8.      } else{
9.      //code to be executed if all given conditions are false
10.     }
```

### Flowchart

Fig: if-else-if Statement



### Example

```
1. <?php
2.     $marks=69;
3.     if ($marks<33){
4.         echo "fail";
5.     }
6.     else if ($marks>=34 && $marks<50) {
7.         echo "D grade";
8.     }
9.     else if ($marks>=50 && $marks<65) {
10.        echo "C grade";
11.    }
12.    else if ($marks>=65 && $marks<80) {
13.        echo "B grade";
14.    }
```



```
15.     else if ($marks>=80 && $marks<90) {
16.         echo "A grade";
17.     }
18.     else if ($marks>=90 && $marks<100) {
19.         echo "A+ grade";
20.     }
21.     else {
22.         echo "Invalid input";
23.     }
24.     ?>
```

### Output:

```
B Grade
```

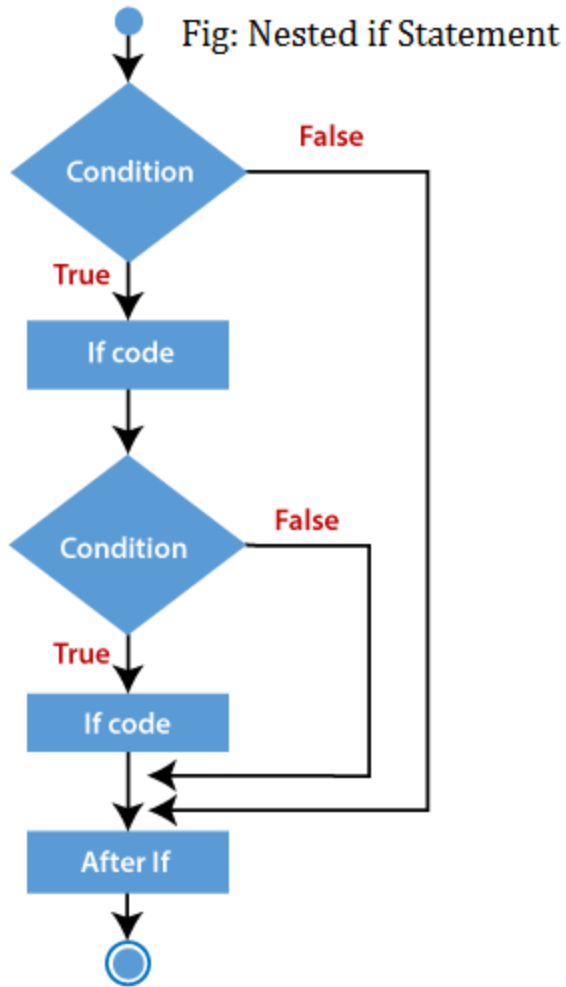
## PHP nested if Statement

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.

### Syntax

```
1.     if (condition) {
2.         //code to be executed if condition is true
3.     if (condition) {
4.         //code to be executed if condition is true
5.     }
6.     }
```

### Flowchart



### Example

```

1.    <?php
2.        $age = 23;
3.        $nationality = "Indian";
4.        //applying conditions on nationality and age
5.        if ($nationality == "Indian")
6.        {
7.            if ($age >= 18) {
8.                echo "Eligible to give vote";
9.            }
10.           else {
11.               echo "Not eligible to give vote";
12.           }
  
```

13.        }
14.        ?>

**Output:**

```
Eligible to give vote
```

**PHP Switch Example**

1.        <?php
2.                \$a = 34; \$b = 56; \$c = 45;
3.        **if** (\$a < \$b) {
4.                **if** (\$a < \$c) {
5.                        echo "\$a is smaller than \$b and \$c";
6.                }
7.        }
8.        ?>

**Output:**

```
34 is smaller than 56 and 45
```

## PHP For Loop

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.

It allows users to put all the loop related statements in one place. See in the syntax given below:

### Syntax

1.        **for**(initialization; condition; increment/decrement){
2.        //code to be executed
3.        }

## Parameters

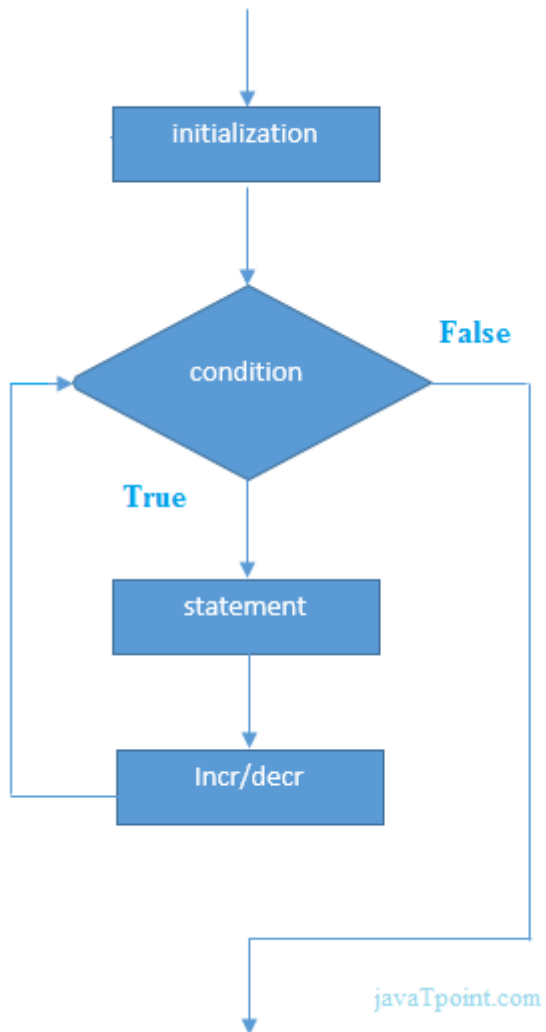
The php for loop is similar to the java/C/C++ for loop. The parameters of for loop have the following meanings:

**initialization** - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.

**condition** - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.

**Increment/decrement** - It increments or decrements the value of the variable.

## Flowchart



## Example

1. `<?php`
2. `for($n=1;$n<=10;$n++){`
3. `echo "$n<br/>";`
4. `}`
5. `?>`

## Output:

```
1
2
3
```

```
4
5
6
7
8
9
10
```

## Example

All three parameters are optional, but semicolon (;) is must to pass in for loop. If we don't pass parameters, it will execute infinite.

```
1.      <?php
2.          $i = 1;
3.          //infinite loop
4.          for (;;) {
5.              echo $i++;
6.              echo "</br>";
7.          }
8.      ?>
```

### Output:

```
1
2
3
4
.
.
.
```

## Example

Below is the example of printing numbers from 1 to 9 in four different ways using for loop.

```
1.      <?php
2.          /* example 1 */
3.
4.          for ($i = 1; $i <= 9; $i++) {
5.              echo $i;
6.          }
```

```
7.      echo "</br>";
8.
9.      /* example 2 */
10.
11.     for ($i = 1;; $i++) {
12.         if ($i > 9) {
13.             break;
14.         }
15.         echo $i;
16.     }
17.     echo "</br>";
18.
19.     /* example 3 */
20.
21.     $i = 1;
22.     for (;) {
23.         if ($i > 9) {
24.             break;
25.         }
26.         echo $i;
27.         $i++;
28.     }
29.     echo "</br>";
30.
31.     /* example 4 */
32.
33.     for ($i = 1, $j = 0; $i <= 9; $j += $i, print $i, $i++);
34.     ?>
```

### Output:

```
123456789
123456789
123456789
123456789
```

## PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop. The inner for loop executes only when the outer for loop condition is found **true**.

In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

### Example

```
1.      <?php
2.      for($i=1;$i<=3;$i++){
3.      for($j=1;$j<=3;$j++){
4.      echo "$i $j<br/>";
5.      }
6.      }
7.      ?>
```

### Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

## PHP For Each Loop

PHP for each loop is used to traverse array elements.

### Syntax

```
1.      foreach( $array as $var ){
2.      //code to be executed
3.      }
```



4.       ?>

### Example

```
1.       <?php
2.       $season=array("summer","winter","spring","autumn");
3.       foreach( $season as $arr ){
4.        echo "Season is: $arr<br />";
5.        }
6.       ?>
```

### Output:

```
Season is: summer
Season is: winter
Season is: spring
Season is: autumn
```

For more details of foreach loop, [click here](#).

---

## PHP foreach loop

The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.

The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.

In foreach loop, we don't need to increment the value.

### Syntax

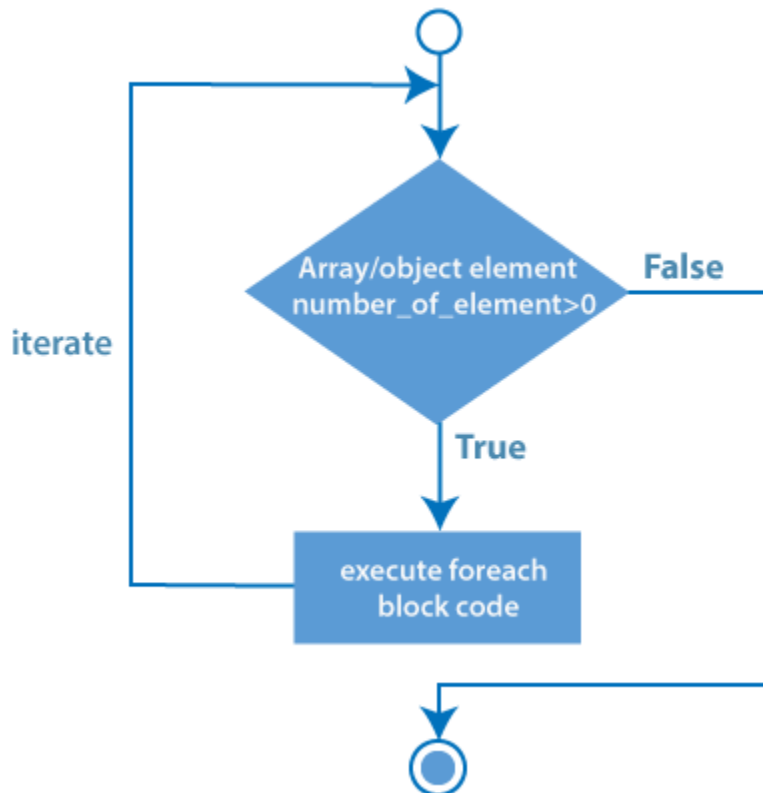
```
1.       foreach ($array as $value) {
2.        //code to be executed
3.        }
```

There is one more syntax of foreach loop.

## Syntax

1. **foreach** (\$array as \$key => \$element) {
2.     //code to be executed
3. }

## Flowchart



## Example 1:

PHP program to print array elements using foreach loop.

1. `<?php`
2.     //declare array
3.     \$season = **array** ("Summer", "Winter", "Autumn", "Rainy");
- 4.
5.     //access array elements using foreach loop
6.     **foreach** (\$season as \$element) {
7.         echo "\$element";
8.         echo "</br>";

9.            }
10.          ?>

**Output:**

```
Summer
Winter
Autumn
Rainy
```

## PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

---

### Advantage of PHP Array

**Less Code:** We don't need to define multiple variables.

**Easy to traverse:** By the help of single loop, we can traverse all the elements of an array.