Operating Systems Syllabus

Unit	Details
I	**Introduction:** What is an operating system? History of operating system, computer hardware, different operating systems, operating system concepts, system calls, operating system structure **Processes and Threads :**Processes, threads, interprocess communication, scheduling, IPC problems.
II	**Memory Management:**No memory abstraction, memory abstraction: address spaces, virtual memory, page replacement algorithms, design issues for paging systems, implementation issues, segmentation. **File Systems:**Files, directories, file system implementation, file-system management and optimization, MS-DOS file system, UNIX V7 file system, CD ROM file system.
III	**Input-Output:** Principles of I/O hardware, Principles of I/O software, I/O software layers, disks, clocks, user interfaces: keyboard, mouse, monitor, thin clients, power management **Deadlocks:**Resources, introduction to deadlocks, the ostrich algorithm, deadlock detection and recovery, deadlock avoidance, deadlock prevention, issues.
IV	**Virtualization and Cloud:**History, requirements for virtualization, type 1 and 2 hypervisors, techniques for efficient virtualization, hypervisor microkernels, memory virtualization, I/O virtualization, Virtual appliances, virtual machines on multicore CPUs, Clouds. **Multiple Processor Systems**Multiprocessors, multicomputers, distributed systems.
V	**Case Study on LINUX and ANDROID:**History of Unix and Linux, Linux Overview, Processes in Linux, Memory management in Linux, I/O in Linux, Linux file system, security in Linux. Android **Case Study on Windows:** History of windows through Windows 10, programming windows, system structure, processes and threads in windows, memory management, caching in windows, I/O in windows, Windows NT file system, Windows power management, Security in windows.

UNIT-1:CHAPTER-1

INTRODUCTION

What is an operating system? History of operating system, computer hardware, different operating systems, operating system concepts, system calls, operating system structure

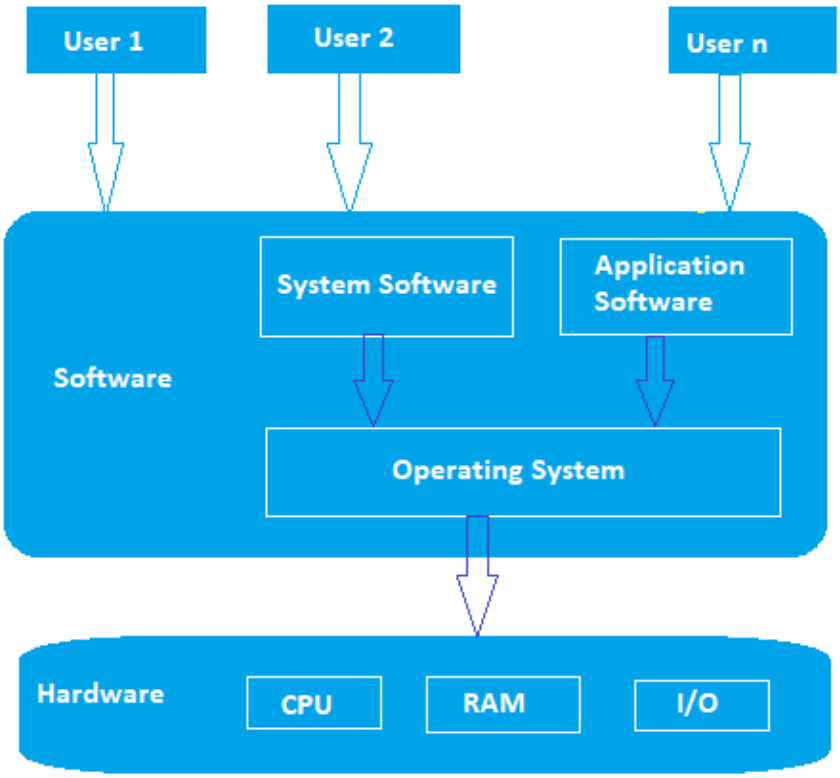
[1.1]What is an operating system?

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

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

Definition

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



Following are some of important functions of an operating System.

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

Memory Management:

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

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

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management:

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

- Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

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

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

- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

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

An Operating System does the following activities for file management –

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

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

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

[1.2] History of operating system,

History Of OS

- Operating systems were first developed in the late 1950s to manage tape storage
- The General Motors Research Lab implemented the first OS in the early 1950s for their IBM 701
- In the mid-1960s, operating systems started to use disks
- In the late 1960s, the first version of the Unix OS was developed

- The first OS built by Microsoft was DOS. It was built in 1981 by purchasing the 86-DOS software from a Seattle company
- The present-day popular OS Windows first came to existence in 1985 when a GUI was created and paired with MS-DOS.

Features of Operating System

Here is a list commonly found important features of an Operating System:

- Protected and supervisor mode
- Allows disk access and file systems Device drivers Networking Security
- Program Execution
- Memory management Virtual Memory Multitasking
- Handling I/O operations
- Manipulation of the file system
- Error Detection and handling
- Resource allocation
- Information and Resource Protection

[1.3]computer hardware,

Hardware represents the physical and tangible components of a computer, i.e. the components that can be seen and touched.

Examples of Hardware are the following –

- **Input devices** – keyboard, mouse, etc.
- **Output devices** – printer, monitor, etc.
- **Secondary storage devices** – Hard disk, CD, DVD, etc.
- **Internal components** – CPU, motherboard, RAM, etc.

Input Device:-

Following are some of the important input devices which are used in a computer –

- Keyboard
- Mouse
- Joy Stick
- Light pen
- Track Ball
- Scanner
- Graphic Tablet
- Microphone
- Magnetic Ink Card Reader(MICR)
- Optical Character Reader(OCR)

- Bar Code Reader
- Optical Mark Reader(OMR)

Keyboard

Keyboard is the most common and very popular input device which helps to input data to the computer. The layout of the keyboard is like that of traditional typewriter, although there are some additional keys provided for performing additional functions.

Mouse

Mouse is the most popular pointing device. It is a very famous cursor-control device having a small palm size box with a round ball at its base, which senses the movement of the mouse and sends corresponding signals to the CPU when the mouse buttons are pressed.

Generally, it has two buttons called the left and the right button and a wheel is present between the buttons. A mouse can be used to control the position of the cursor on the screen, but it cannot be used to enter text into the computer.

Advantages

- Easy to use
- Not very expensive
- Moves the cursor faster than the arrow keys of the keyboard.

Joystick

Joystick is also a pointing device, which is used to move the cursor position on a monitor screen. It is a stick having a spherical ball at its both lower and upper ends. The lower spherical ball moves in a socket. The joystick can be moved in all four directions.

The function of the joystick is similar to that of a mouse. It is mainly used in Computer Aided Designing (CAD) and playing computer games.

Light Pen

Light pen is a pointing device similar to a pen. It is used to select a displayed menu item or draw pictures on the monitor screen. It consists of a photocell and an optical system placed in a small tube

When the tip of a light pen is moved over the monitor screen and the pen button is pressed, its photocell sensing element detects the screen location and sends the corresponding signal to the CPU.

Track Ball

Track ball is an input device that is mostly used in notebook or laptop computer, instead of a mouse. This is a ball which is half inserted and by moving fingers on the ball, the pointer can be moved.

Since the whole device is not moved, a track ball requires less space than a mouse. A track ball comes in various shapes like a ball, a button, or a square.

Scanner

Scanner is an input device, which works more like a photocopy machine. It is used when some information is available on paper and it is to be transferred to the hard disk of the computer for further manipulation.

Scanner captures images from the source which are then converted into a digital form that can be stored on the disk. These images can be edited before they are printed.

Digitizer

Digitizer is an input device which converts analog information into digital form. Digitizer can convert a signal from the television or camera into a series of numbers that could be stored in a computer. They can be used by the computer to create a picture of whatever the camera had been pointed at.

Digitizer is also known as Tablet or Graphics Tablet as it converts graphics and pictorial data into binary inputs. A graphic tablet as digitizer is used for fine works of drawing and image manipulation applications.

Microphone

Microphone is an input device to input sound that is then stored in a digital form.

The microphone is used for various applications such as adding sound to a multimedia presentation or for mixing music.

Magnetic Ink Card Reader (MICR)

MICR input device is generally used in banks as there are large number of cheques to be processed every day. The bank's code number and cheque number are printed on the cheques with a special type of ink that contains particles of magnetic material that are machine readable.

Output Device:-

Following are some of the important output devices used in a computer.

- Monitors
- Graphic Plotter
- Printer

Monitors

Monitors, commonly called as **Visual Display Unit** (VDU), are the main output device of a computer. It forms images from tiny dots, called pixels that are arranged in a rectangular form. The sharpness of the image depends upon the number of pixels.

There are two kinds of viewing screen used for monitors.

- Cathode-Ray Tube (CRT)
- Flat-Panel Display

Cathode-Ray Tube (CRT) Monitor

The CRT display is made up of small picture elements called pixels. The smaller the pixels, the better the image clarity or resolution. It takes more than one illuminated pixel to form a whole character, such as the letter 'e' in the word help.

A finite number of characters can be displayed on a screen at once. The screen can be divided into a series of character boxes - fixed location on the screen where a standard character can be placed. Most screens are capable of displaying 80 characters of data horizontally and 25 lines vertically.

There are some disadvantages of CRT –

- Large in Size
- High power consumption

Flat-Panel Display Monitor

The flat-panel display refers to a class of video devices that have reduced volume, weight and power requirement in comparison to the CRT. You can hang them on walls or wear them on your wrists. Current uses of flat-panel displays include calculators, video games, monitors, laptop computer, and graphics display.

The flat-panel display is divided into two categories –

- **Emissive Displays** – Emissive displays are devices that convert electrical energy into light. For example, plasma panel and LED (Light-Emitting Diodes).
- **Non-Emissive Displays** – Non-emissive displays use optical effects to convert sunlight or light from some other source into graphics patterns. For example, LCD (Liquid-Crystal Device).

Printers

Printer is an output device, which is used to print information on paper.

There are two types of printers –

- Impact Printers
- Non-Impact Printers

Impact Printers

Impact printers print the characters by striking them on the ribbon, which is then pressed on the paper.

Characteristics of Impact Printers are the following –

- Very low consumable costs
- Very noisy
- Useful for bulk printing due to low cost
- There is physical contact with the paper to produce an image

These printers are of two types –

- Character printers
- Line printers

Character Printers

Character printers are the printers which print one character at a time.

These are further divided into two types:

- Dot Matrix Printer(DMP)
- Daisy Wheel

Dot Matrix Printer

In the market, one of the most popular printers is Dot Matrix Printer. These printers are popular because of their ease of printing and economical price. Each character printed is in the form of pattern of dots and head consists of a Matrix of Pins of size (5*7, 7*9, 9*7 or 9*9) which come out to form a character which is why it is called Dot Matrix Printer.

Advantages

- Inexpensive
- Widely Used
- Other language characters can be printed

Disadvantages

- Slow Speed
- Poor Quality

Daisy Wheel

Head is lying on a wheel and pins corresponding to characters are like petals of Daisy (flower) which is why it is called Daisy Wheel Printer. These printers are generally used for word-processing in offices that require a few letters to be sent here and there with very nice quality.

Relationship between Hardware and Software

- Hardware and software are mutually dependent on each other. Both of them must work together to make a computer produce a useful output.
- Software cannot be utilized without supporting hardware.
- Hardware without a set of programs to operate upon cannot be utilized and is useless.
- To get a particular job done on the computer, relevant software should be loaded into the hardware.
- Hardware is a one-time expense.
- Software development is very expensive and is a continuing expense.
- Different software applications can be loaded on a hardware to run different jobs.
- A software acts as an interface between the user and the hardware.
- If the hardware is the 'heart' of a computer system, then the software is its 'soul'. Both are complementary to each other.

[1.4]different operating systems,

An Operating System performs all the basic tasks like managing file,process, and memory. Thus operating system acts as manager of all the resources, i.e. **resource manager**. Thus operating system becomes an interface between user and machine.
Types of Operating Systems: Some of the widely used operating systems are as follows-

1.Batch Operating System –

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

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

Disadvantages of Batch Operating System:

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

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

2. Time-Sharing Operating Systems –

Each task is given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The

time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.

Advantages of Time-Sharing OS:

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

Disadvantages of Time-Sharing OS:

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

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

3. Distributed Operating System –

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

Advantages of Distributed Operating System:

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

Disadvantages of Distributed Operating System:

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

Examples of Distributed Operating System are- LOCUS etc

4. Network Operating System –

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

A **dvantages of Network Operating System:**

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

Disadvantages of Network Operating System:

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

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

5.Real-Time Operating System –

These types of OSs serves the real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

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

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

- **Hard Real-Time Systems:**
  These OSs are meant for the applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or air bags which are required to be readily available in case of any accident. Virtual memory is almost never found in these systems.
- **Soft Real-Time Systems:**
  These OSs are for applications where for time-constraint is less strict.

Advantages of RTOS:

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

Disadvantages of RTOS:

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

- **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.

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

[1.5]SYSTEM CALLS:-

Introduction of System Call:

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls :

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.

Types of System Calls : There are 5 different categories of system calls –

1. **Process control:** end, abort, create, terminate, allocate and free memory.
2. **File management:** create, open, close, delete, read file etc.
3. Device management
4. Information maintenance
5. Communication

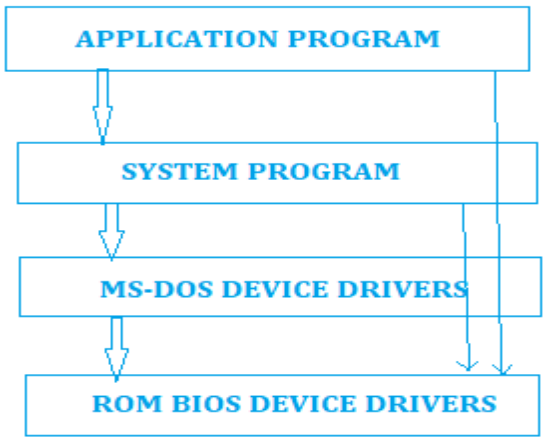
[1.6] operating system structure:

An operating system is a construct that allows the user application programs to interact with the system hardware. Since the operating system is such a complex structure, it should be created with utmost care so it can be used and modified easily. An easy way to do this is to create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

Simple Structure

There are many operating systems that have a rather simple structure. These started as small systems and rapidly expanded much further than their scope. A common example of this is MS-DOS. It was designed simply for a niche amount for people. There was no indication that it would become so popular.

An image to illustrate the structure of MS-DOS is as follows –



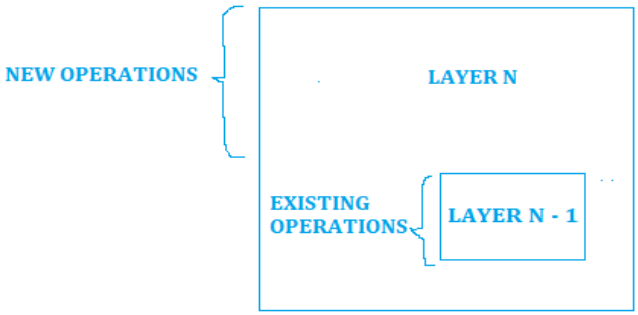
MS-DOS STRUCTURE

It is better that operating systems have a modular structure, unlike MS-DOS. That would lead to greater control over the computer system and its various applications. The modular structure would also allow the programmers to hide information as required and implement internal routines as they see fit without changing the outer specifications.

Layered Structure:

One way to achieve modularity in the operating system is the layered approach. In this, the bottom layer is the hardware and the topmost layer is the user interface.

An image demonstrating the layered approach is as follows –



LAYERED STRUCTURE OF OPERATING SYSTEM

UNIT-1:CHAPTER-2

Processes and Threads:

[2.1]Processes,

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A process is an 'active' entity, as opposed to a program, which is considered to be a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

Text Section :A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the ***Program Counter***.
Stack: The Stack contains the temporary data, such as function parameters, returns addresses, and local variables.
Data Section: Contains the global variable.
Heap Section: Dynamically allocated memory to process during its run time.
Refer this for more details on sections.

A process has following attributes\Characteristic:

1. Process Id: A unique identifier assigned by the operating system

2. Process State: Can be ready, running, etc.

3. CPU registers: Like the Program Counter (CPU registers must be saved and restored when a process is swapped in and out of CPU)

4. Accounts information:

5. I/O status information: For example, devices allocated to the process, open files, etc

6. CPU scheduling information: For example, Priority (Different processes may have different priorities, for example a short process may be assigned a low priority in the shortest job first scheduling)

All of the above attributes of a process are also known as the ***context of the process***. Every process has its own program control block(PCB), i.e each process will have a unique PCB. All of the above attributes are part of the PCB.

States of Process:

A process is in one of the following states:

1. New: Newly Created Process (or) being-created process.

2. Ready: After creation process moves to Ready state, i.e. the process is ready for execution.

3. Run: Currently running process in CPU (only one process at a time can be under execution in a single processor).

4. Wait (or Block): When a process requests I/O access.

5. Complete (or Terminated): The process completed its execution.

6. Suspended Ready: When the ready queue becomes full, some processes are moved to suspended ready state

7. Suspended Block: When waiting queue becomes full.

Context Switching:

The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from running state to ready state.

When does context switching happen?

1. When a high-priority process comes to ready state (i.e. with higher priority than the running process)
2. An Interrupt occurs
3. User and kernel mode switch (It is not necessary though)
4. Primitive CPU scheduling used.

Context Switch vs Mode Switch

A mode switch occurs when CPU privilege level is changed, for example when a system call is made or a fault occurs. The kernel works in more a privileged mode than a standard user task. If a user process wants to access things which are only accessible to the kernel, a mode switch must occur. The currently executing process need not be changed during a mode switch.
A mode switch typically occurs for a process context switch to occur. Only the kernel can cause a context switch.

CPU-Bound vs I/O-Bound Processes:

A CPU-bound process requires more CPU time or spends more time in the running state. An I/O-bound process requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.

[2.2] Threads,

What is a Thread?
A thread is a path of execution within a process. A process can contain multiple threads.
Why Multithreading?
A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces. Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Advantages of Thread over Process

1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.
2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.
3. . Effective utilization of multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.
4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.
Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.
5. Communication: Communication between multiple threads is easier, as the threads shares common address space. while in process we have to follow some specific communication technique for communication between two process.
6. Enhanced throughput of the system: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.
Types of Threads
There are two types of threads.
User Level Thread
Kernel Level Thread

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing

message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

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

Kernel Level Threads

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

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

Advantages

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

Disadvantages

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

Multithreading Models:

Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors

and a blocking system call need not block the entire process. Multithreading models are three types

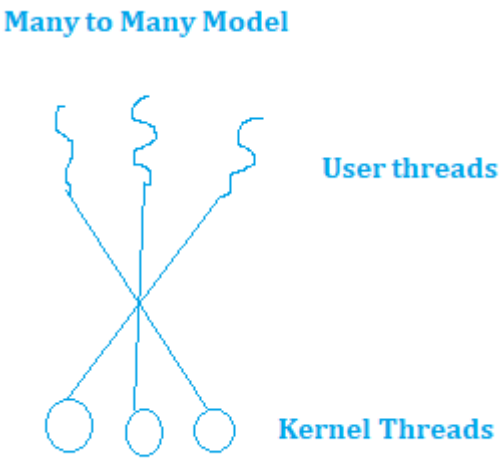
- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

The following diagram shows the many-to-many threading model where 6 user level threads are multiplexing with 6 kernel level threads. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine. This model provides the best accuracy on concurrency and when a thread performs a blocking system call, the kernel can schedule another thread for execution.

Diagram:

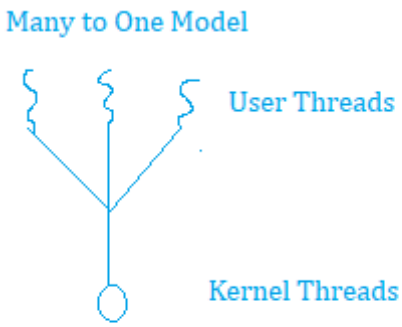


Many to One Model

Many-to-one model maps many user level threads to one Kernel-level thread. Thread management is done in user space by the thread library. When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

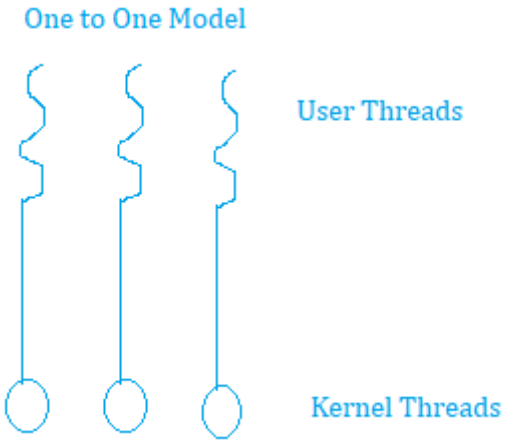
If the user-level thread libraries are implemented in the operating system in such a way that the system does not support them, then the Kernel threads use the many-to-one relationship modes

Diagram:



One to One Model

There is one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.



Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship model.

Multithreading and Multitasking:

S.NO	MULTITASKING	MULTITHREADING
1.	In multitasking, users are	While in multithreading, many

S.NO	MULTITASKING	MULTITHREADING
	allowed to perform many tasks by CPU.	threads are created from a process through which computer power is increased.
2.	Multitasking involves often CPU switching between the tasks.	While in multithreading also, CPU switching is often involved between the threads.
3.	In multitasking, the processes share separate memory.	While in multithreading, processes are allocated same memory.
4.	Multitasking component involves multiprocessing.	While multithreading component does not involve multiprocessing.
5.	In multitasking, CPU is provided in order to execute many tasks at a time.	While in multithreading also, CPU is provided in order to execute many threads from a process at a time.
6.	In multitasking, processes don't share same resources, each process is allocated separate resources.	While in multithreading, each process share same resources.
7.	Multitasking is slow compared to multithreading.	While multithreading is faster.
8.	In multitasking, termination of	While in multithreading, termination

S.NO	MULTITASKING	MULTITHREADING
	process takes more time.	of thread takes less time.

[2.3] Interprocess Communication,

here are numerous reasons for providing an environment or situation which allows process co-operation:

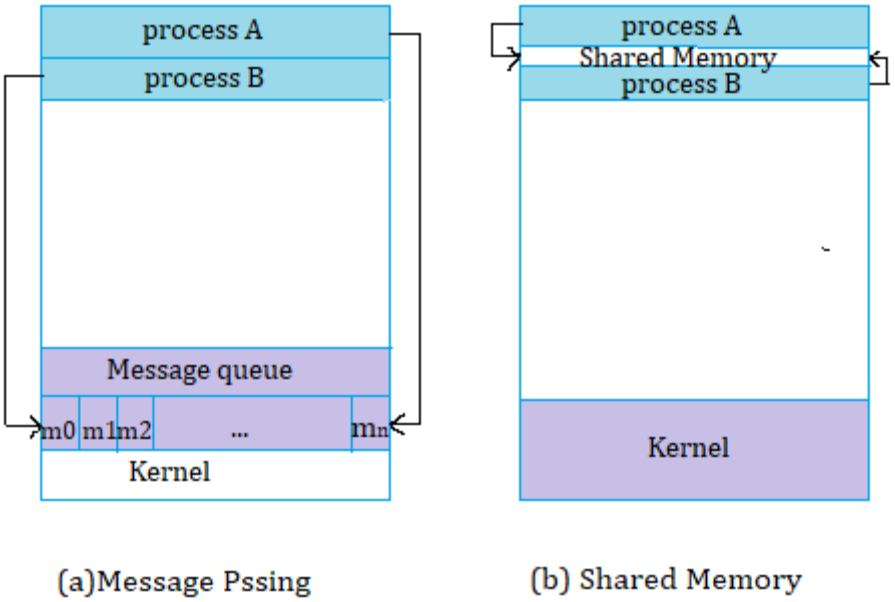
- Information sharing: Since some users may be interested in the same piece of information (for example, a shared file), you must provide a situation for allowing concurrent access to that information.
- Computation speedup: If you want a particular work to run fast, you must break it into sub-tasks where each of them will get executed in parallel with the other tasks. Note that such a speed-up can be attained only when the computer has compound or various processing elements like CPUs or I/O channels.
- Modularity: You may want to build the system in a modular way by dividing the system functions into split processes or threads.
- Convenience: Even a single user may work on many tasks at a time. For example, a user may be editing, formatting, printing, and compiling in parallel.

Working together with multiple processes, require an interprocess communication (IPC) method which will allow them to exchange data along with various information. There are two primary models of interprocess communication:

1. shared memory and
2. message passing.

In the shared-memory model, a region of memory which is shared by cooperating processes gets established. Processes can be then able to exchange information by reading and writing all the data to the shared region. In the message-passing form, communication takes place by way of messages exchanged among the cooperating processes.

The two communications models are contrasted in the figure below:



(a)Message Pssing (b) Shared Memory

Shared Memory Systems

Interprocess communication (IPC) usually utilizes shared memory that requires communicating processes for establishing a region of shared memory. Typically, a shared-memory region resides within the address space of any process creating the shared memory segment. Other processes that wish for communicating using this shared-memory segment must connect it to their address space.

More on Inter Process Shared Memory

Note that, normally what happens, the operating system tries to check one process from accessing other's process's memory. Shared memory needs that two or more processes agree to remove this limitation. They can then exchange information via reading and writing data within the shared areas.
The form of the data and the location gets established by these processes and are not under the control of the operating system. The processes are also in charge to ensure that they are not writing to the same old location simultaneously.

[2.4]Scheduling, Ipc Problems

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

- First-Come, First-Served (FCFS) Scheduling

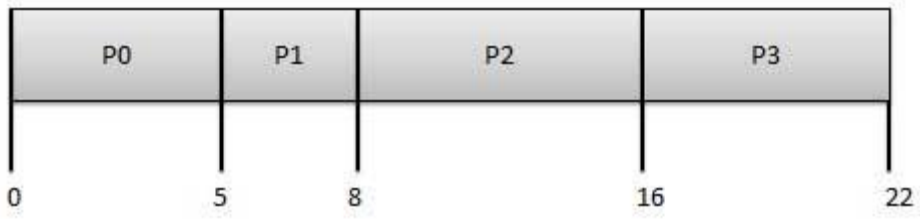
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

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

First Come First Serve (FCFS)

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

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



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4

P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

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

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

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

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0

P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

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

Priority Based Scheduling

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

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

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

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0

P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

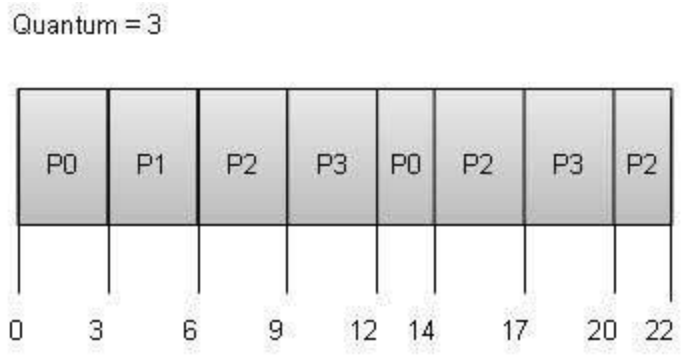
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9

P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

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

Multiple-Level Queues Scheduling

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

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

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

Unit-2:Chapter-1

Memory Management: No memory abstraction, memory abstraction: address spaces, virtual memory, page replacement algorithms, design issues for paging systems, implementation issues, segmentation.

.

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

This tutorial will teach you basic concepts related to Memory Management.

Process Address Space:

The process address space is the set of logical addresses that a process references in its code. For example, when 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; that is, 2^31 possible numbers, for a total theoretical size of 2 gigabytes.

The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program. There are three types of addresses used in a program before and after memory is allocated −

S.N.	Memory Addresses & Description
1	**Symbolic addresses** The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.
2	**Relative addresses** At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3	**Physical addresses** The loader generates these addresses at the time when a program is loaded into main memory.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a **logical address space**. The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space.**

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.

Static vs Dynamic Loading:

The choice between Static or Dynamic Loading is to be made at the time of computer program being developed. If you have to load your program statically, then at the time of compilation, the complete programs will be compiled and linked without leaving any external program or module dependency. The linker combines the object program with other necessary object modules into an absolute program, which also includes logical addresses.

If you are writing a Dynamically loaded program, then your compiler will compile the program and for all the modules which you want to include dynamically, only references will be provided and rest of the work will be done at the time of execution.

At the time of loading, with **static loading**, the absolute program (and data) is loaded into memory in order for execution to start.

If you are using **dynamic loading**, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

Static vs Dynamic Linking:

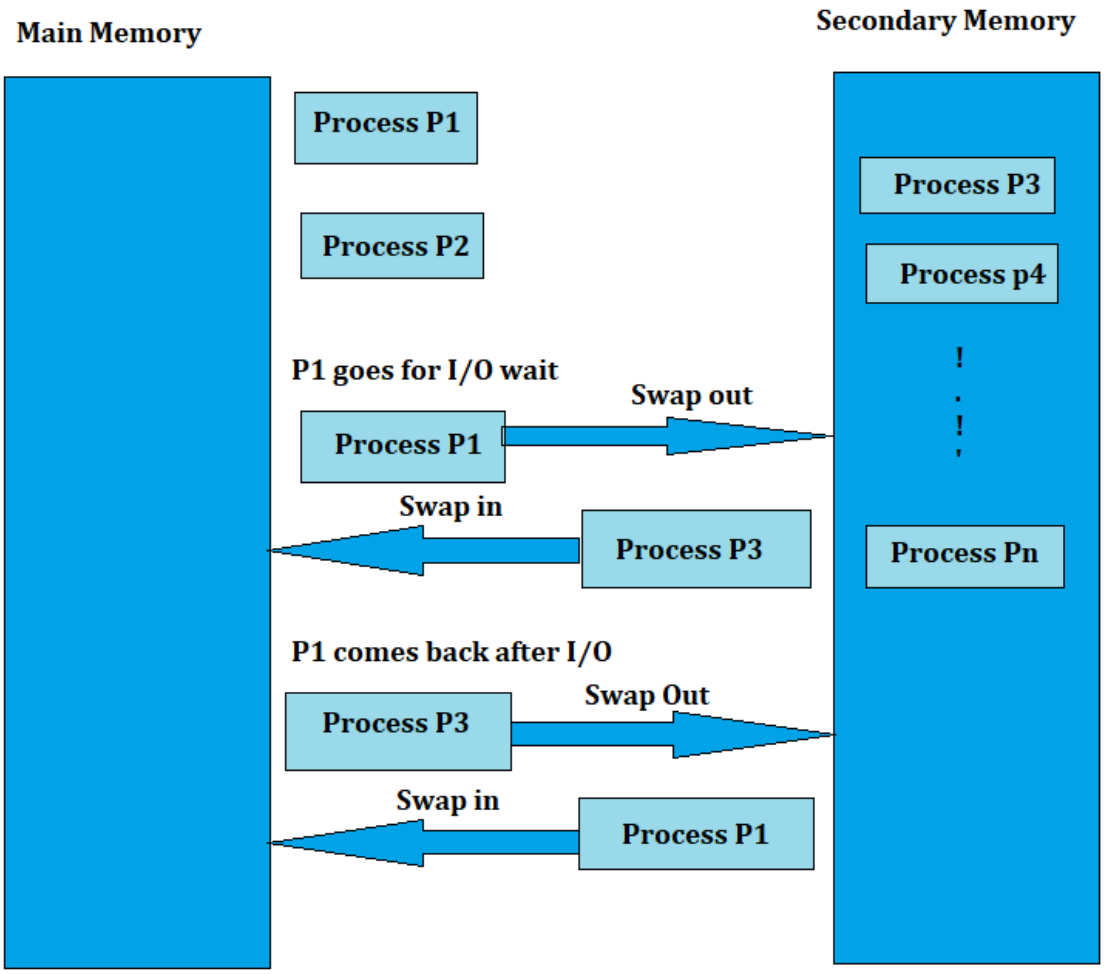
As explained above, when static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.

When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking. Dynamic Link Libraries (DLL) in Windows and Shared Objects in Unix are good examples of dynamic libraries.

Swapping:

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

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048KB / 1024KB per second
= 2 seconds
= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Memory Allocation:

Main memory usually has two partitions –

- **Low Memory** – Operating system resides in this memory.
- **High Memory** – User processes are held in high memory.

Operating system uses the following memory allocation mechanism.

S.N.	Memory Allocation & Description
1	**Single-partition allocation** In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
2	**Multiple-partition allocation** In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

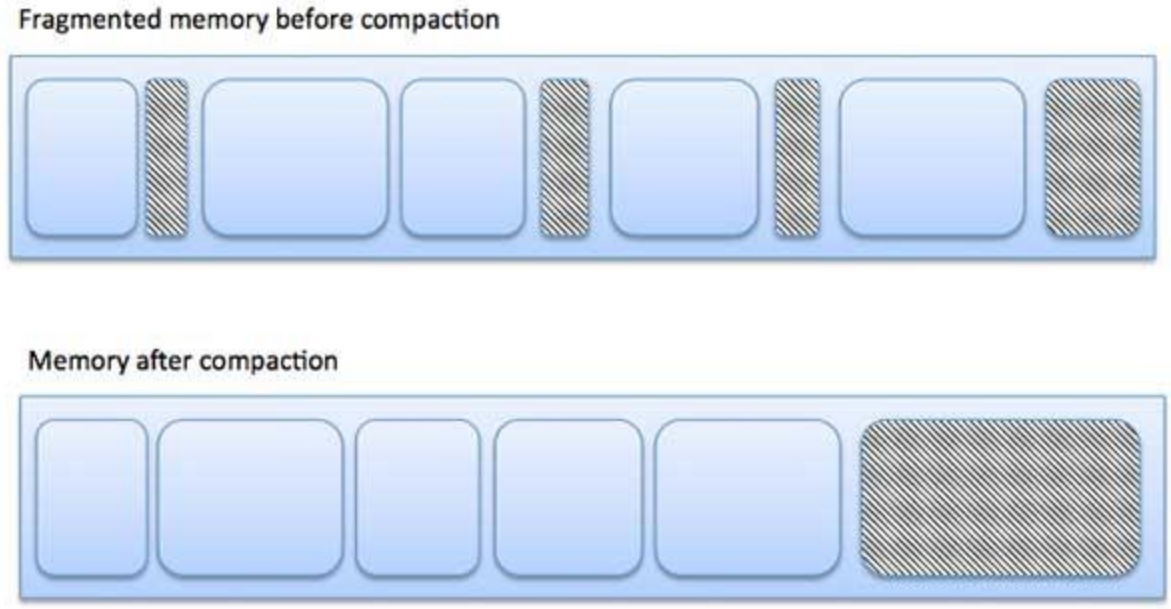
Fragmentation:

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

Fragmentation is of two types –

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

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



External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

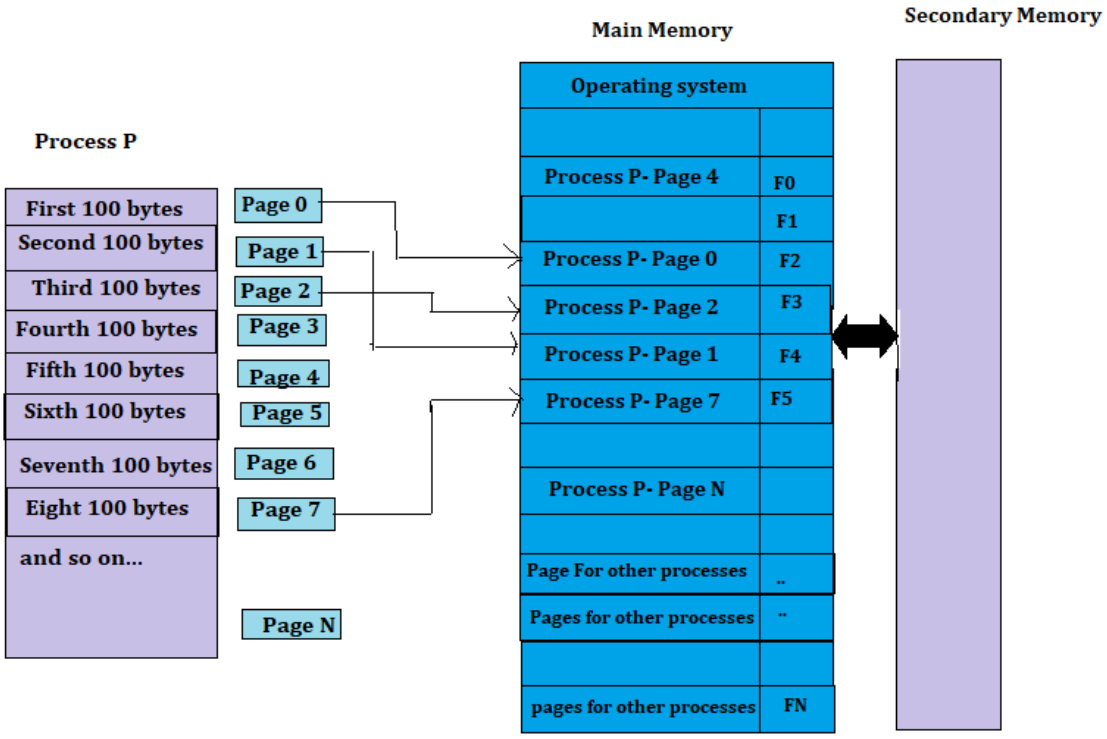
The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Paging:

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Address Translation

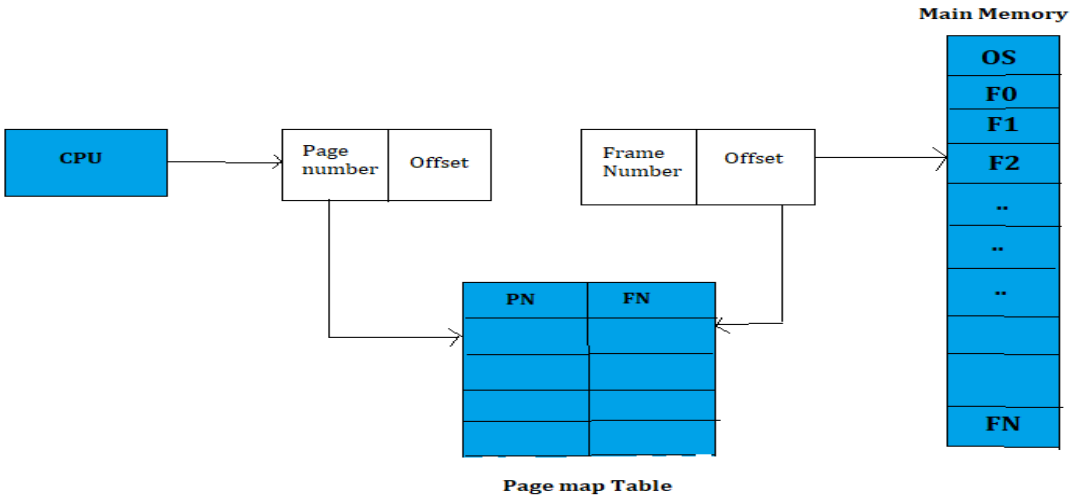
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

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

Physical Address = Frame number + page offset

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



When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging

Here is a list of advantages and disadvantages of paging –

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

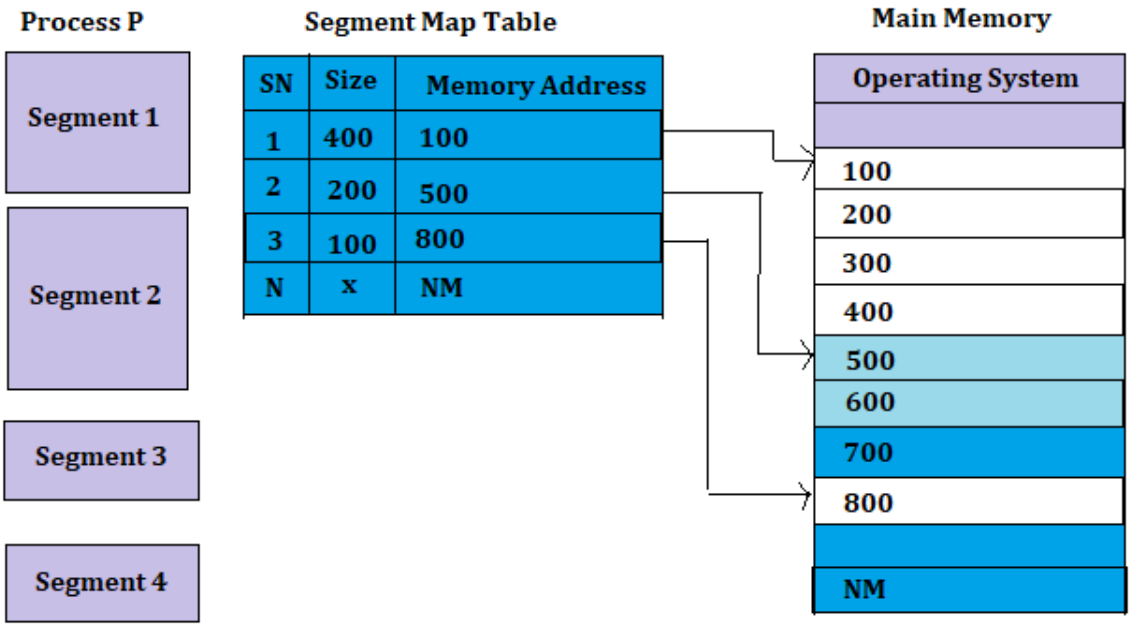
Segmentation:

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Virtual memory:

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

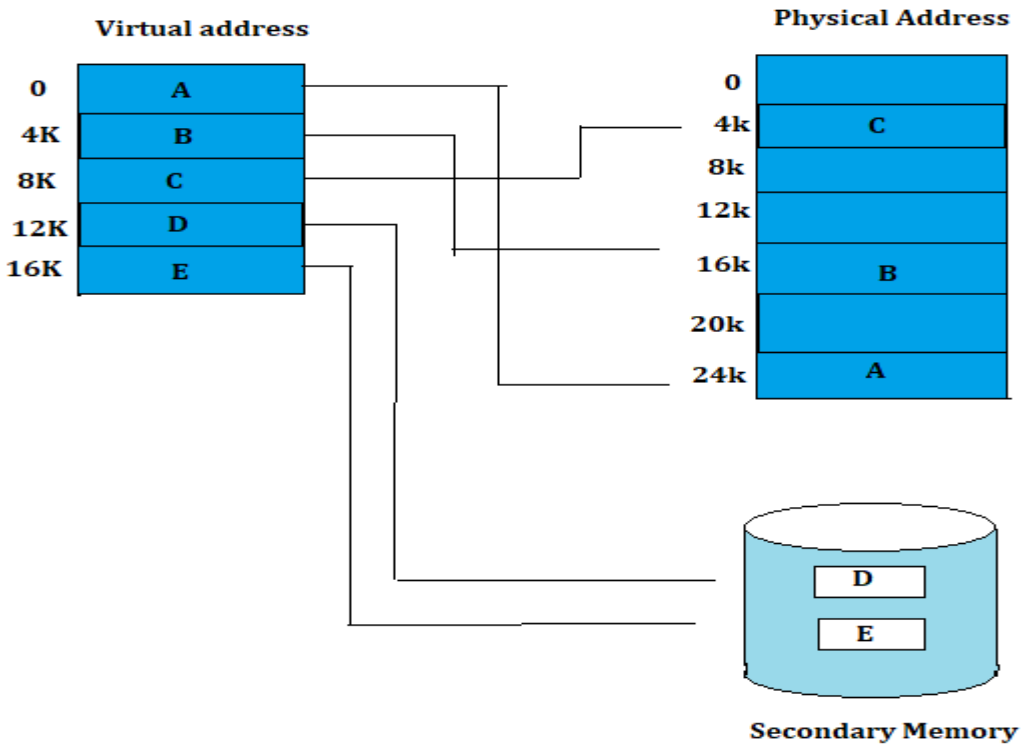
The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.

- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

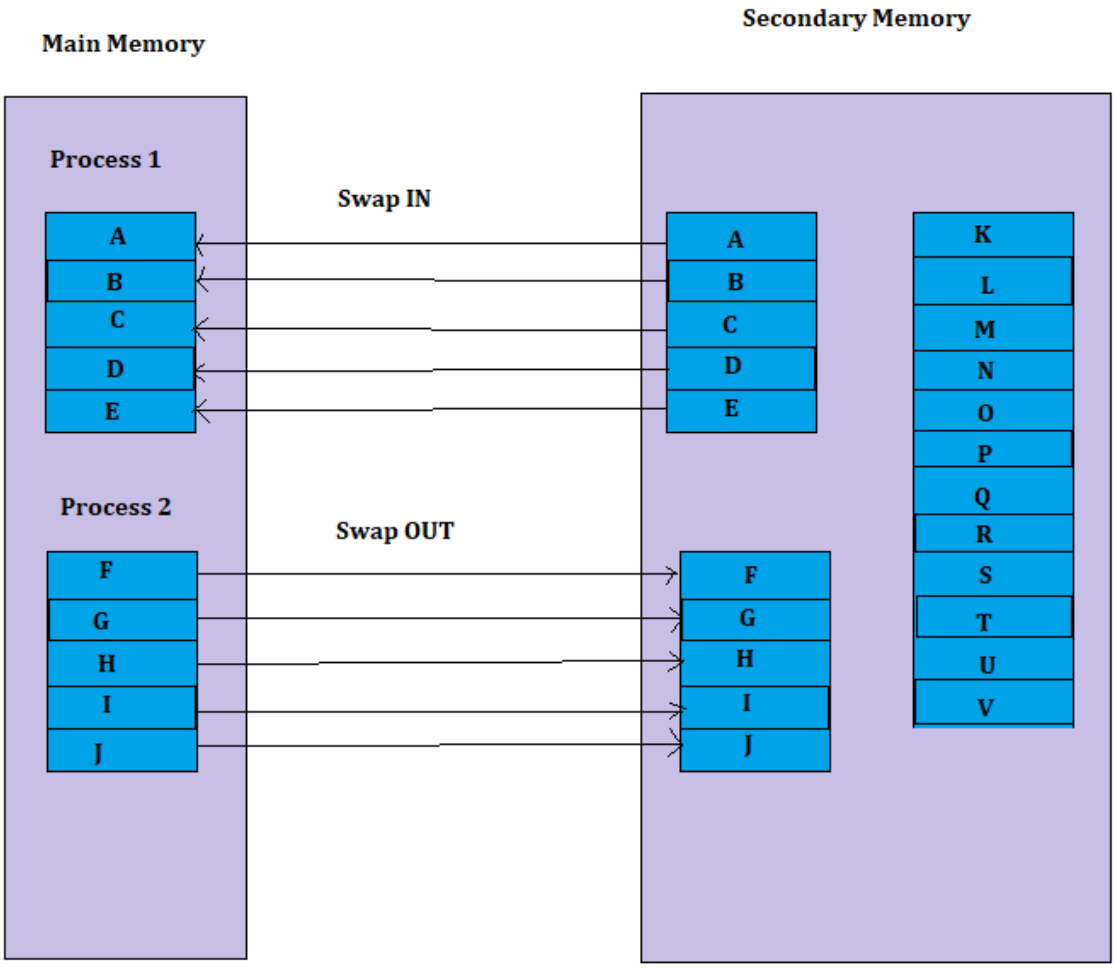
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below –



Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Demand Paging:

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



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

Advantages

Following are the advantages of Demand Paging –

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

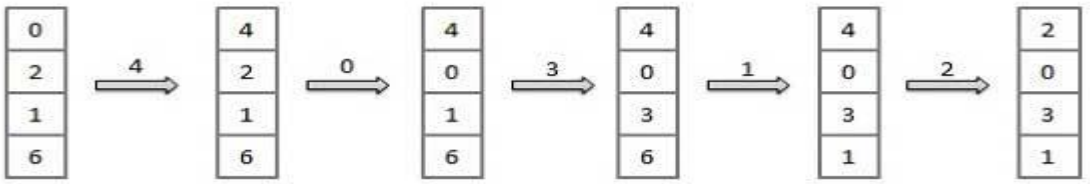
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses – 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x x



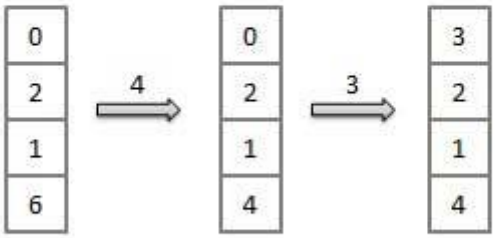
Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



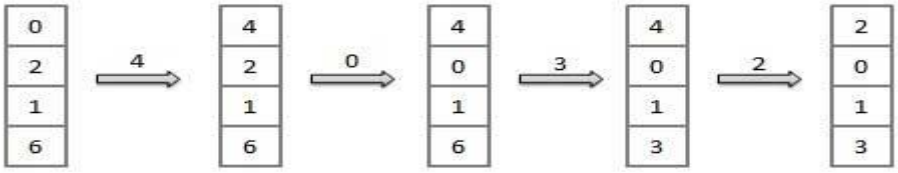
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

UNIT-2:CHATPER-2

File Systems: Files, directories, file system implementation, file-system management and optimization, MS-DOS file system, UNIX V7 file system, CD ROM file system

FILE DIRECTORIES:

Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Information contained in a device directory are:

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

Operation performed on directory are:

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

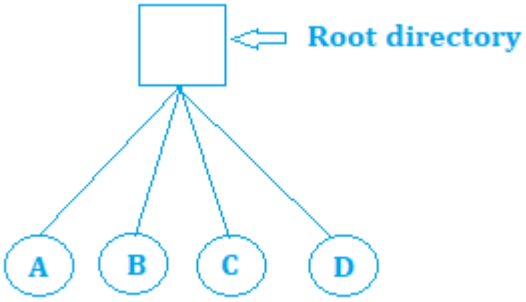
Advantages of maintaining directories are:

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

SINGLE-LEVEL DIRECTORY

In this a single directory is maintained for all the users.

- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.



TWO-LEVEL DIRECTORY

In this separate directories for each user is maintained.

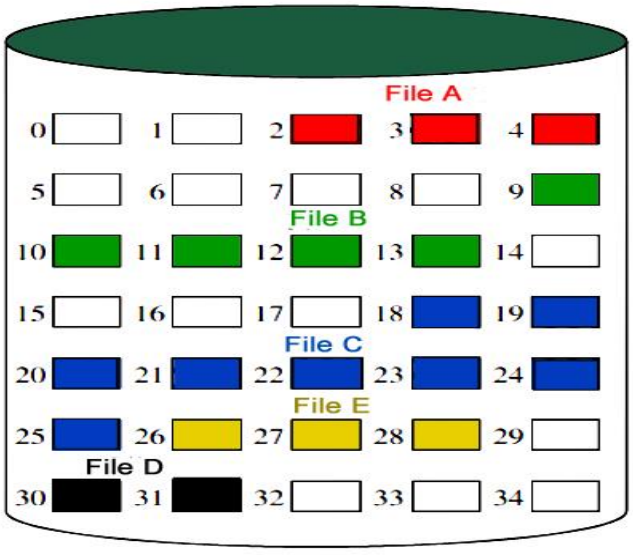
- Path name:Due to two levels there is a path name for every file to locate that file.
- Now,we can have same file name for different user.
- Searching is efficient in this method.

TREE-STRUCTURED DIRECTORY:

Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.

FILE ALLOCATION METHODS:-

1. Continuous Allocation: A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block. For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



File allocation table

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

Disadvantage

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

2. Linked Allocation(Non-contiguous allocation) : Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

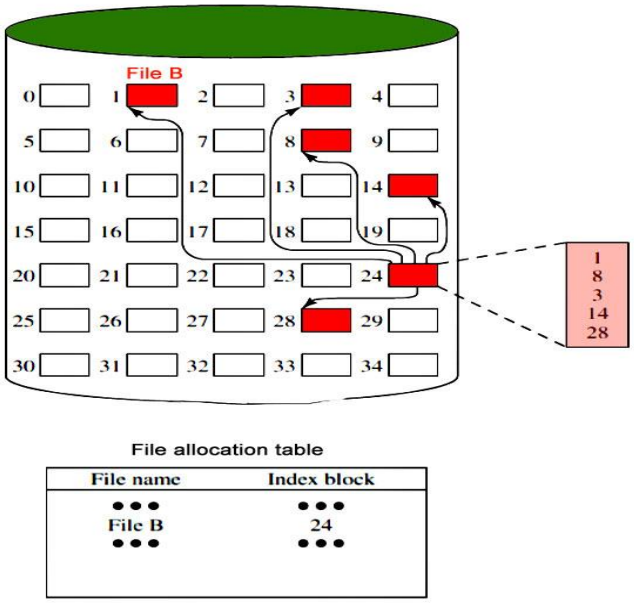
Disadvantage:

- Internal fragmentation exists in last disk block of file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.
- It supports only the sequencial access of files.

3.Indexed Allocation:

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form

of file allocation.



File allocation table

File name	Index block
File B	24

File Syatem Management and Optimization:

Disk-Space Management

Since all the files are normally stored on disk one of the main concerns of file system is management of disk space.

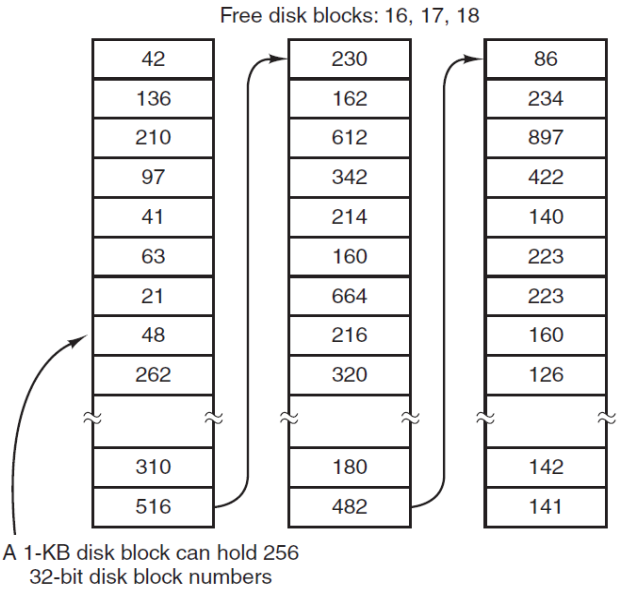
Block Size

The main question that arises while storing files in a fixed-size blocks is the size of the block. If the block is too large space gets wasted and if the block is too small time gets waste. So, to choose a correct block size some information about the file-size distribution is required. Performance and space-utilization are always in conflict.

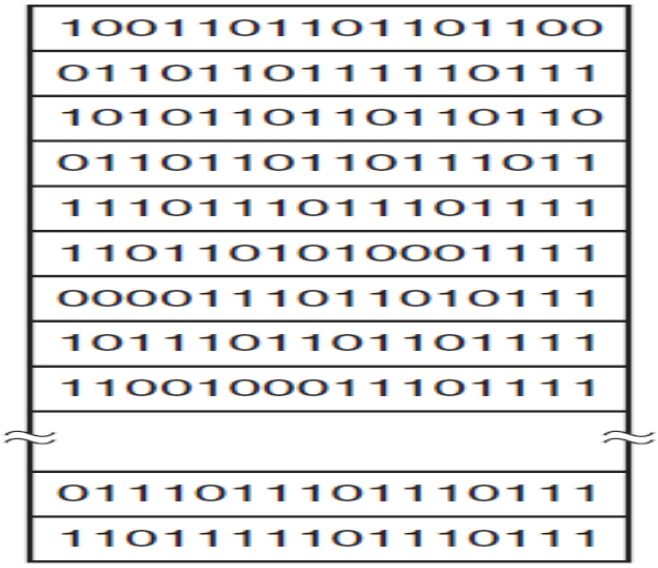
Keeping track of free blocks

After a block size has been finalized the next issue that needs to be catered is how to keep track of the free blocks. In order to keep track there are two methods that are widely used:

- Using a linked list: Using a linked list of disk blocks with each block holding as many free disk block numbers as will fit.



- Bitmap: A disk with n blocks has a bitmap with n bits. Free blocks are represented using 1's and allocated blocks as 0's as seen below in the figure.



Disk quotas

Multiuser operating systems often provide a mechanism for enforcing disk quotas. A system administrator assigns each user a maximum allotment of files and blocks and the operating system makes sure that the users do not exceed their quotas. Quotas are kept track of on a per-user basis in a quota table.

File-system Backups

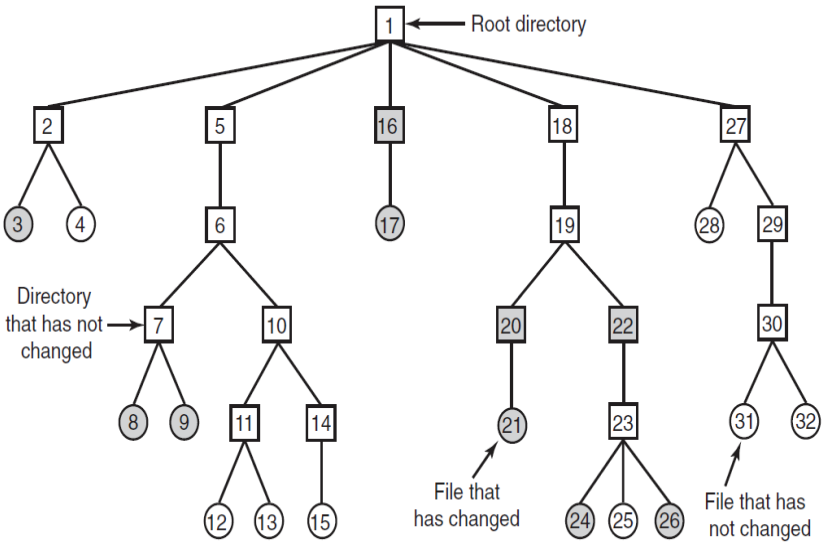
If a computer's file system is irrevocably lost, whether due to hardware or software restoring all the information will be difficult, time consuming and in many cases impossible. So it is advised to always have file-system backups.

Backing up files is time consuming and as well occupies large amount of space, so doing it efficiently and conveniently is important. Below are few points to be considered before creating backups for files.

- Is it required to backup the entire file system or only a part of it.
- Backing up files that haven't been changed from previous backup leads to **incremental dumps**. So it's better to take a backup of only those files which have changed from the time of previous backup. But recovery gets complicated in such cases.
- Since there is immense amount of data, it is generally desired to compress the data before taking a backup for the same.
- It is difficult to perform a backup on an active file-system since the backup may be inconsistent.
- Making backups introduces many security issues

There are two ways for dumping a disk to the backup disk:

- **Physical dump:** In this way dump starts at block 0 of the disk, writes all the disk blocks onto thee output disk in order and stops after copying the last one. **Advantages:** Simplicity and great speed. **Disadvantages:** inability to skip selected directories, make incremental dumps, and restore individual files upon request
- **Logical dump:** In this way the dump starts at one or more specified directories and recursively dump all files and directories found that have been changed since some given base date. This is the most commonly used way.



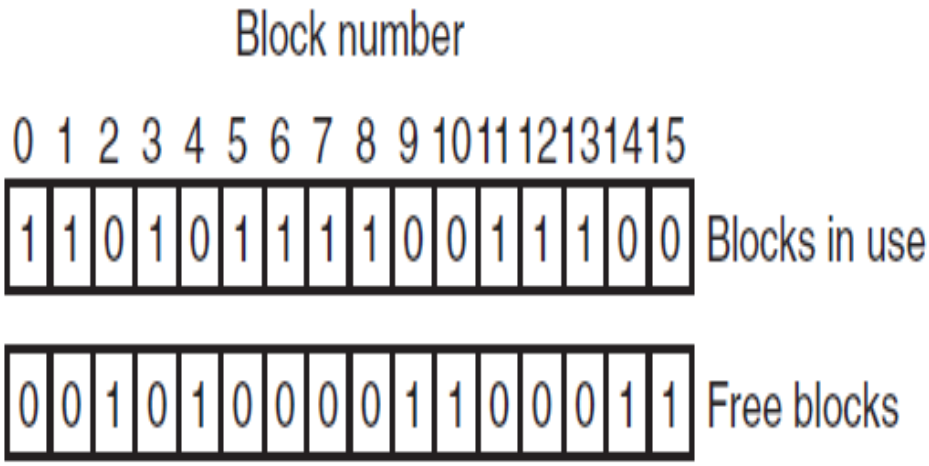
The above figure depicts a popular algorithm used in many UNIX systems wherein squares depict directories and circles depict files. This algorithm dumps all the files and directories that have been modified and also the ones on the path to a modified file or directory. The dump algorithm maintains a bitmap indexed by i-node number with several bits per i-node. Bits will be set and cleared in this map as the algorithm proceeds. Although logical dumping is straightforward, there are few issues associated with it.

- Since the free block list is not a file, it is not dumped and hence it must be reconstructed from scratch after all the dumps have been restored
- If a file is linked to two or more directories, it is important that the file is restored only one time and that all the directories that are supposed to point to it do so
- UNIX files may contain holes
- Special files, named pipes and all other files that are not real should never be dumped.

File-system Consistency:

To deal with inconsistent file systems, most computers have a utility program that checks file-system consistency. For example, UNIX has fsck and Windows has sfc. This utility can be run whenever the system is booted. The utility programs perform two kinds of consistency checks.

- Blocks: To check block consistency the program builds two tables, each one containing a counter for each block, initially set to 0. If the file system is consistent, each block will have a 1 either in the first table or in the second table as you can see in the figure below.



In case if both the tables have 0 in it that may be because the block is missing and hence will be reported as a missing block. The two other situations are if a block is seen more than once in free list and same data block is present in two or more files.

- In addition to checking to see that each block is properly accounted for, the file-system checker also checks the directory system. It too uses a table of counters but per file-size rather than per block. These counts start at 1 when a file is created and are incremented each time a (hard) link is made to the file. In a consistent file system, both counts will agree

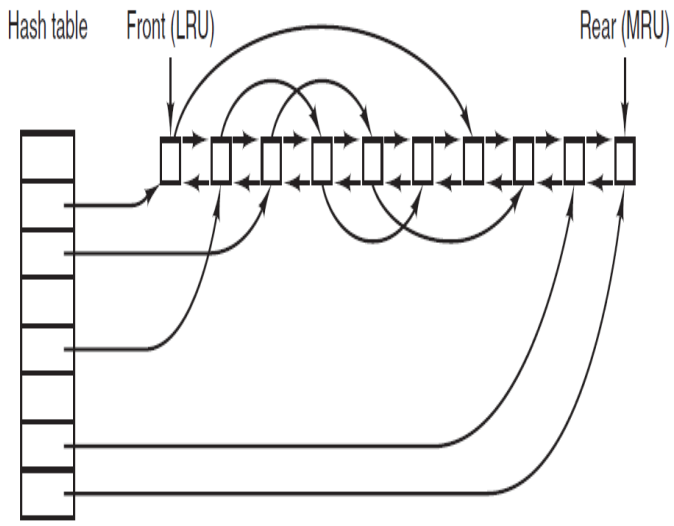
File-system Performance:

Since the access to disk is much slower than access to memory, many file systems have been designed with various optimizations to improve performance as described below.

Caching:

The most common technique used to reduce disk access time is the block cache or buffer cache. Cache can be defined as a collection of items of the same type stored in a hidden or inaccessible place. The most common algorithm for cache works in such a way that if a disk access is initiated,

the cache is checked first to see if the disk block is present. If yes then the read request can be satisfied without a disk access else the disk block is copied to cache first and then the read request is processed.



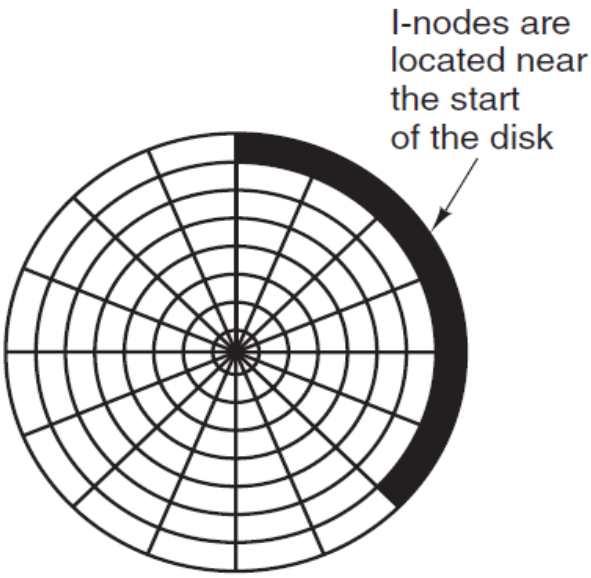
The above figure depicts how to quickly determine if a block is present in a cache or not. For doing so a hash table can be implemented and look up the result in a hash table.

Block Read Ahead

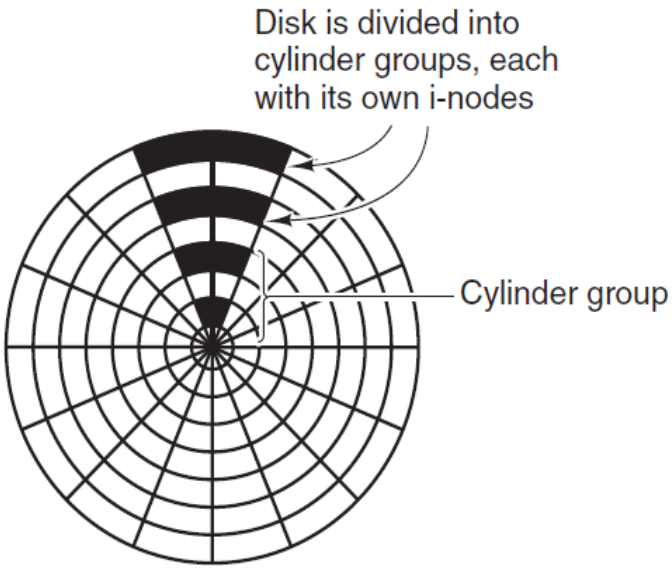
Another technique to improve file-system performance is to try to get blocks into the cache before they are needed to increase the hit rate. This works only when files are read sequentially. When a file system is asked for block 'k' in the file it does that and then also checks before hand if 'k+1' is available if not it schedules a read for the block k+1 thinking that it might be of use later.

Reducing disk arm motion

Another way to increase file-system performance is by reducing the disk-arm motion by putting blocks that are likely to be accessed in sequence close to each other ,preferably in the same cylinder.



In the above figure all the i-nodes are near the start of the disk, so the average distance between an inode and its blocks will be half the number of cylinders, requiring long seeks. But to increase the performance the placement of i-nodes can be modified as below.



Defragmenting Disks

Due to continuous creation and removal of files the disks get badly fragmented with files and holes all over the place. As a consequence, when a new file is created, the blocks used for it may be spread all over the disk, giving poor performance. The performance can be restored by moving files around to make them contiguous and to put all (or at least most) of the free space in one or more large contiguous regions on the disk.

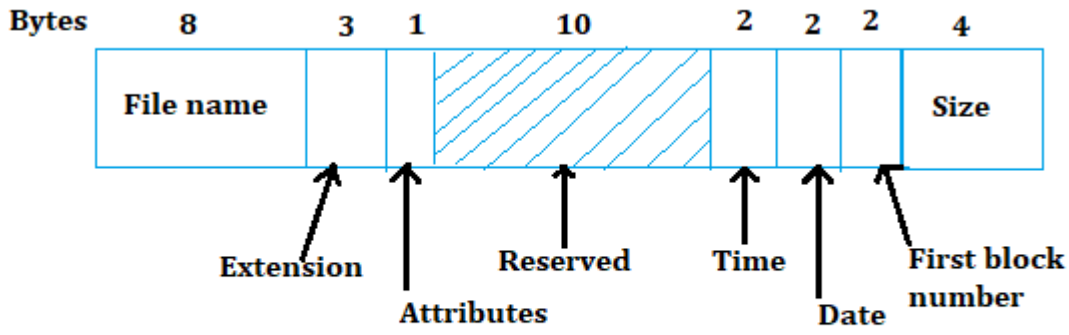
MS-DOS FILE SYSTEM:

1. The ms-dos file system is the one the first ibm pcs came with .it was the main file system up through windows 98 and windows me. It is still supported on windows 2000 , windows xp, and windows vista, although it is no longer standard on new pcs now except for floppy disks.

2.however, it and an extension of it (fat -32)have become widely used for many embedded systems. Most digital cameras use it . Many mp3 players use it exclusively. The popular apple ipad uses it as the default file system, although knowledgeable hackers can reformat the ipad and install a different file system.

3.thus the numebr of electronic devices using the ms-dos file system is vastly larger now than at any time in the past, and certainly much larger than the number using the more modern ntfs file system.

4.to read a file , an ms-dos program must first make an open system call to get a handle for it. The open system call specifies a path ,which may be either absolute or relative to the the current working directory.the path is looked up component by component until the final direcotory is located and read into memory.it is then searched for the file to be opened .



MS-DOS directory entry

UNIX V7 File System:

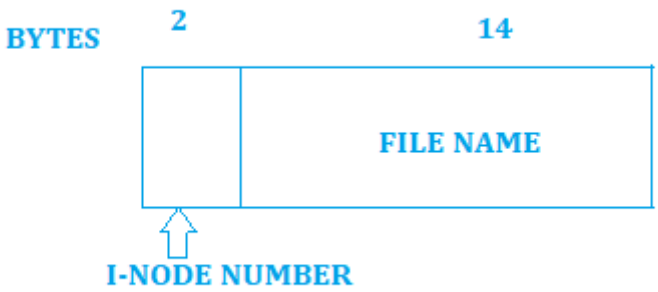
In this file system each directory entry contains a name and a pointer to the corresponding i-node. This metadata for or directory is stored in the corresponding I-node. The early UNIX limited file names to 14 characters, stored in a fixed length field .The name field now is of varying length and file name can be quite long. Here the directory hierarchy takes two steps: get the I-node, get the file or subdirectory. This shows how important it is not to parse filenames for each I/O operation, i.e. why the open() system call is important.

UNIX V7 file system was used in PDP-11 minicomputers .File are placed in a tree structure having a root directory. New items which are added to the root directory in the tree are known as links .Here , file names can be at a maximum of 14 characters. ASCII characters can be used in the file name except the forward slash(/) since it is used as a separator in path names.

UNIX V7 directory entry: Here are the computers of a UNIX V7 directory entry:

1) I-node number: It is used to represent the number of I-node for a file. It has a size of 2 bytes. I-nodes has attributes including file size, information about creation ,last access and last modification of file ,group, protection information, owner, and count of number of directory entries that point to i-node. whenever a link is added, count is increased by a and whenever a link is removed ,count is decreased by 1.

2)File name :It represents the file name .It has a size of 14 bytes.



CD ROM file system:

CD-ROM File Systems

As our first example of a file system, let us consider the file systems used on CD-ROMs. These systems are particularly simple because they were designed for write-once media. Among other things, for example, they have no provision for keeping track of free blocks because on a CD-ROM files cannot be freed or added after the disk has been manufactured. Below we will take a look at the main CD-ROM file system type and two extensions to it.

The ISO 9660 File System

The most common standard for CD-ROM file systems was adopted as an International Standard in 1988 under the name ISO 9660. Virtually every CD-ROM currently on the market is compatible with this standard, sometimes with the extensions to be discussed below. One of the goals of this standard was to make every CD-ROM readable on every computer, independent of the byte ordering used and independent of the operating

system used. As a consequence, some limitations were placed on the file system to make it possible for the weakest operating systems then in use (such as MS-DOS) to read it.

CD-ROMs do not have concentric cylinders the way magnetic disks do. Instead there is a single continuous spiral containing the bits in a linear sequence (although seeks across the spiral are possible). The bits along the spiral are divided into logical blocks (also called logical sectors) of 2352 bytes. Some of these are for preambles, error correction, and other overhead. The payload portion of each logical block is 2048 bytes. When used for music, CDs have leadins, leadouts, and intertrack gaps, but these are not used for data CD-ROMs. Often the position of a block along the spiral is quoted in minutes and seconds. It can be converted to a linear block number using the conversion factor of 1 sec = 75 blocks.

ISO 9660 supports CD-ROM sets with as many as $2^{16} - 1$ CDs in the set. The individual CD-ROMs may also be partitioned into logical volumes (partitions). However, below we will concentrate on ISO 9660 for a single unpartitioned CD-ROM.

Every CD-ROM begins with 16 blocks whose function is not defined by the ISO 9660 standard. A CD-ROM manufacturer could use this area for providing a bootstrap program to allow the computer to be booted from the CD-ROM, or for some other purpose. Next comes one block containing the **primary volume descriptor**, which contains some general information about the CD-ROM. Among this information are the system identifier (32 bytes), volume identifier (32 bytes), publisher identifier (128 bytes), and data preparer identifier (128 bytes). The manufacturer can fill in these fields in any desired way, except that only upper case letters, digits, and a very small number of punctuation marks may be used to ensure cross-platform compatibility.

The primary volume descriptor also contains the names of three files, which may contain the abstract, copyright notice, and bibliographic information, respectively. In addition, certain key numbers are also present, including the logical block size (normally 2048, but 4096, 8192, and larger powers of two are allowed in certain cases), the number of blocks on the CD-ROM, and the creation and expiration dates of the CD-ROM. Finally, the primary volume descriptor also contains a directory entry for the root directory, telling where to find it on the CD-ROM (i.e., which block it starts at). From this directory, the rest of the file system can be located.

In addition to the primary volume descriptor, a CD-ROM may contain a supplementary volume descriptor. It contains similar information to the primary, but that will not concern us here.

The root directory, and all other directories for that matter, consists of a variable number of entries, the last of which contains a bit marking it as the final one. The directory entries themselves are also variable length. Each directory entry consists of 10 to 12 fields, some of which are in ASCII and others of which are numerical fields in binary. The binary fields are encoded twice, once in little-endian format (used on example). Thus a 16-bit number uses 4 bytes and a 32-bit number uses 8 bytes. The use of this redundant coding was necessary to avoid hurting anyone's feelings when the standard was developed. If the standard had dictated little endian, then people from companies with big-endian products would have felt like second-class citizens and would not have accepted the standard. The emotional content of a CD-ROM can thus be quantified and measured exactly in kilobytes/hour of wasted space.

.Directory entries may optionally have an extended attributes. If this feature is used for a directory entry, the second byte tells how long the extended attributes are.

Next comes the starting block of the file itself. Files are stored as contiguous runs of blocks, so a file's location is completely specified by the starting block and the size, which is contained in the next field.

The date and time that the CD-ROM was recorded is stored in the next field, with separate bytes for the year, month, day, hour, minute, second, and time zone. Years begin to count at 1900, which means that CD-ROMs will suffer from a Y2156 problem because the year following 2155 will be 1900. This problem could have been delayed by defining the origin of time to be 1988 (the year the standard was adopted). Had that been done, the problem would have been postponed until 2244. Every 88 extra years helps.

The Flags field contains a few miscellaneous bits, including one to hide the entry in listings (a feature copied from MS-DOS), one to distinguish an entry that is a file from an entry that is a directory, one to enable the use of the extended attributes, and one to mark the last entry in a directory. A few other bits are also present in this field but they will not concern us here. The next field deals with interleaving pieces of files in a way that is not used in the simplest version of ISO 9660, so we will not consider it further.

The next field tells which CD-ROM the file is located on. It is permitted that a directory entry on one CD-ROM refers to a file located on another CD-ROM in the set. In this way it is possible to build a master directory on the first CD-ROM that lists all the files on all the CD-ROMs in the complete set.

The field marked gives the size of the file name in bytes. It is followed by the file name itself. A file name consists of a base name, a dot, an extension, a semicolon, and a binary version number (1 or 2 bytes). The base name and extension may use upper case letters, the digits 0–9, and the underscore character. All other characters are forbidden to make sure that every computer can handle every file name. The base name can be up to eight characters; the extension can be up to three characters. These choices were dictated by the need to be MS-DOS compatible. A file name may be present in a directory multiple times, as long as each one has a different version number.

The last two fields are not always present. The Padding field is used to force every directory entry to be an even number of bytes, to align the numeric fields of subsequent entries on 2-byte boundaries. If padding is needed, a 0 byte is used. Finally, we have the System use field. Its function and size are undefined, except that it must be an even number of bytes. Different systems use it in different ways. The Macintosh keeps Finder flags here, for example.

Entries within a directory are listed in alphabetical order except for the first two entries. The first entry is for the directory itself. The second one is for its parent. In this respect, these entries are similar to the UNIX . and .. directory entries. The files themselves need not be in directory order.

There is no explicit limit to the number of entries in a directory. However, there is a limit to the depth of nesting. The maximum depth of directory nesting is eight.

ISO 9660 defines what are called three levels. Level 1 is the most restrictive and specifies that file names are limited to 8 + 3 characters as we have described, and also requires all files to be contiguous as we have described. Furthermore, it specifies that directory names be limited to eight characters with no extensions. Use of this level maximizes the chances that a CD-ROM can be read on every computer.

Level 2 relaxes the length restriction. It allows files and directories to have names of up to 31 characters, but still from the same set of characters.

Level 3 uses the same name limits as level 2, but partially relaxes the assumption that files have to be contiguous. With this level, a file may consist of several sections, each of which is a contiguous run of blocks. The same run may occur multiple times in a file and may also occur in two or more files. If large chunks of data are repeated in several files, level 3 provides some space optimization by not requiring the data to be present multiple times.

Rock Ridge Extensions:

As we have seen, ISO 9660 is highly restrictive in several ways. Shortly after it came out, people in the UNIX community began working on an extension to make it possible to represent UNIX file systems on a CD-ROM. These extensions were named Rock Ridge, after a town in the Gene Wilder movie Blazing Saddles, probably because one of the committee members liked the film.

The extensions use the System use field in order to make Rock Ridge CD-ROMs readable on any computer. All the other fields retain their normal ISO 9660 meaning. Any system not aware of the Rock Ridge extensions just ignores them and sees a normal CD-ROM.

The extensions are divided up into the following fields:

1. PX - POSIX attributes.
2. PN - Major and minor device numbers.
3. SL - Symbolic link.
4. NM - Alternative name.
5. CL - Child location.
6. PL - Parent location.
7. RE - Relocation.
8. TF - Time stamps.

The PX field contains the standard UNIX permission bits for the owner, group, and others. It also contains the other bits contained in the mode word, such as the SETUID and SETGID bits, and so on.

To allow raw devices to be represented on a CD-ROM, the PN field is present. It contains the major and minor device numbers associated with the file. In this way, the contents of the /dev directory can be written to a CD-ROM and later reconstructed correctly on the target system.

The SL field is for symbolic links. It allows a file on one file system to refer to a file on a different file system.

Probably the most important field is NM. It allows a second name to be associated with the file. This name is not subject to the character set or length restrictions of ISO 9660, making it possible to express arbitrary UNIX file names on a CD-ROM.

The next three fields are used together to get around the ISO 9660 limit of directories that may only be nested eight deep. Using them it is possible to specify that a directory is to be relocated, and to tell where it goes in the hierarchy. It is effectively a way to work around the artificial depth limit.

Finally, the TF field contains the three timestamps included in each UNIX i-node, namely the time the file was created, the time it was last modified, and the time it was last accessed. Together, these extensions make it possible to copy a UNIX file system to a CD-ROM and then restore it correctly to a different system.

Joliet Extensions

The UNIX community was not the only group that wanted a way to extend ISO 9660. Microsoft also found it too restrictive (although it was Microsoft's own MS-DOS that caused most of the restrictions in the first place). Therefore Microsoft invented some extensions that were called **Joliet**. They were designed to allow Windows file systems to be copied to CD-ROM and then restored, in precisely the same way that Rock Ridge was designed for UNIX. Virtually all programs that run under Windows and use CD-ROMs support Joliet, including programs that burn CD-recordable. Usually, these programs offer a choice between the various ISO 9660 levels and Joliet.

The major extensions provided by Joliet are

1. Long file names.
2. Unicode character set.
3. Directory nesting deeper than eight levels.
4. Directory names with extensions

The first extension allows file names up to 64 characters. The second extension enables the use of the Unicode character set for file names. This extension is important for software intended for use in countries that do not use the Latin alphabet, such as Japan, Israel, and Greece. Since Unicode characters are two bytes, the maximum file name in Joliet occupies 128 bytes.

Like Rock Ridge, the limitation on directory nesting is removed by Joliet. Directories can be nested as deeply as needed. Finally, directory names can have extensions. It is not

clear why this extension was included, since Windows directories virtually never use extensions, but maybe some day they will.

UNIT 3:CHAPTER 1

Input-Output: Principles of I/O hardware, Principles of I/O software, I/O software layers, disks, clocks, user interfaces: keyboard, mouse, monitor, thin clients, power management

Deadlocks: Resources, introduction to deadlocks, the ostrich algorithm, deadlock detection and recovery, deadlock avoidance, deadlock prevention, issues.

Principles of I/O hardware:

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories –

- **Block devices** – A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
- **Character devices** – A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sounds cards etc

Device Controllers:

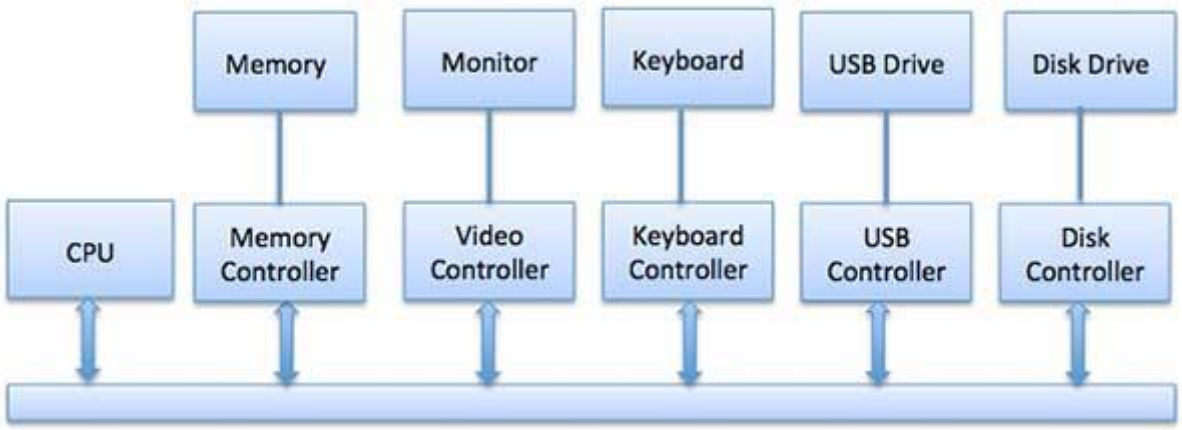
Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU,

memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.



Synchronous vs asynchronous I/O

- **Synchronous I/O** – In this scheme CPU execution waits while I/O proceeds
- **Asynchronous I/O** – I/O proceeds concurrently with CPU execution

Communication to I/O Devices

The CPU must have a way to pass information to and from an I/O device. There are three approaches available to communicate with the CPU and Device.

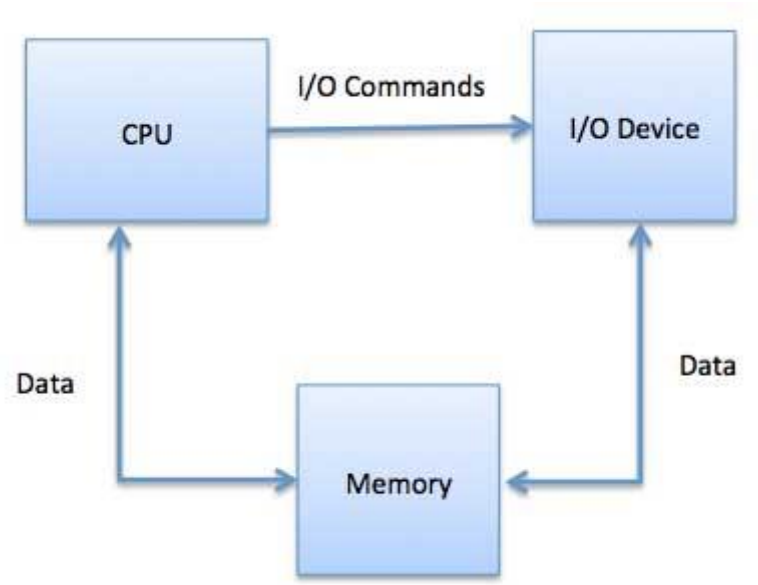
- Special Instruction I/O
- Memory-mapped I/O
- Direct memory access (DMA)

Special Instruction I/O

This uses CPU instructions that are specifically made for controlling I/O devices. These instructions typically allow data to be sent to an I/O device or read from an I/O device.

Memory-mapped I/O

When using memory-mapped I/O, the same address space is shared by memory and I/O devices. The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.



While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished.

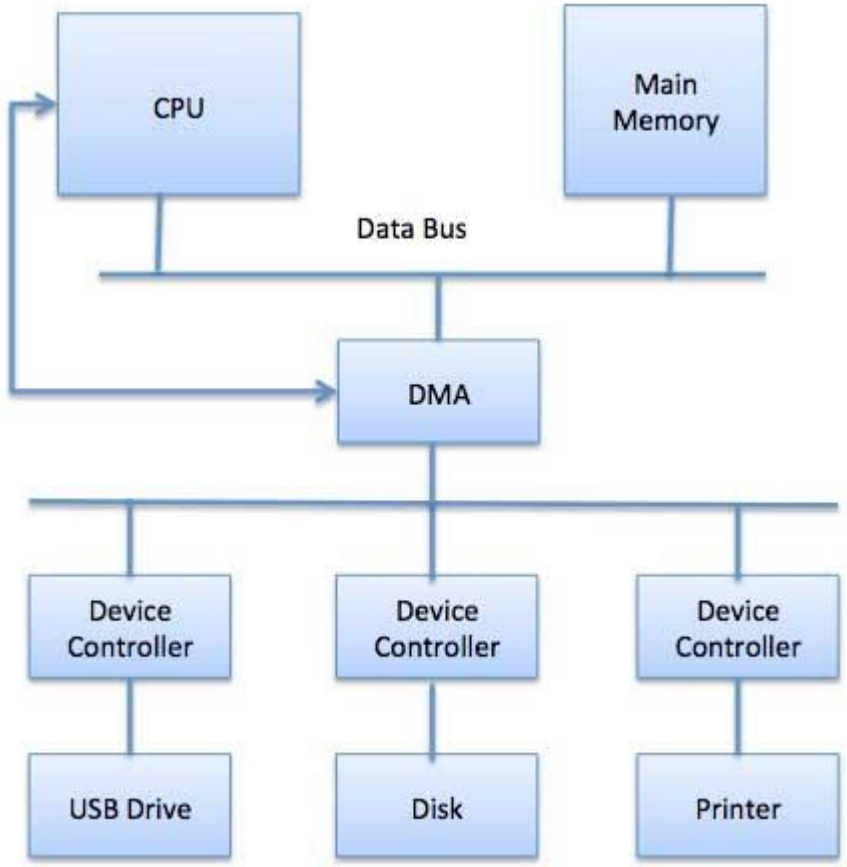
The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped IO is used for most high-speed I/O devices like disks, communication interfaces.

Direct Memory Access (DMA):

Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses direct memory access (DMA) hardware to reduce this overhead.

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.



The operating system uses the DMA hardware as follows –

Step	Description
1	Device driver is instructed to transfer disk data to a buffer address X.
2	Device driver then instruct disk controller to transfer data to buffer.
3	Disk controller starts DMA transfer.
4	Disk controller sends each byte to DMA controller.
5	DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero.

6	When C becomes zero, DMA interrupts CPU to signal transfer completion.

Polling vs Interrupts I/O

A computer must have a way of detecting the arrival of any type of input. There are two ways that this can happen, known as **polling** and **interrupts**. Both of these techniques allow the processor to deal with events that can happen at any time and that are not related to the process it is currently running.

Polling I/O

Polling is the simplest way for an I/O device to communicate with the processor. The process of periodically checking status of the device to see if it is time for the next I/O operation, is called polling. The I/O device simply puts the information in a Status register, and the processor must come and get the information.

Most of the time, devices will not require attention and when one does it will have to wait until it is next interrogated by the polling program. This is an inefficient method and much of the processors time is wasted on unnecessary polls.

Compare this method to a teacher continually asking every student in a class, one after another, if they need help. Obviously the more efficient method would be for a student to inform the teacher whenever they require assistance.

Interrupts I/O

An alternative scheme for dealing with I/O is the interrupt-driven method. An interrupt is a signal to the microprocessor from a device that requires attention.

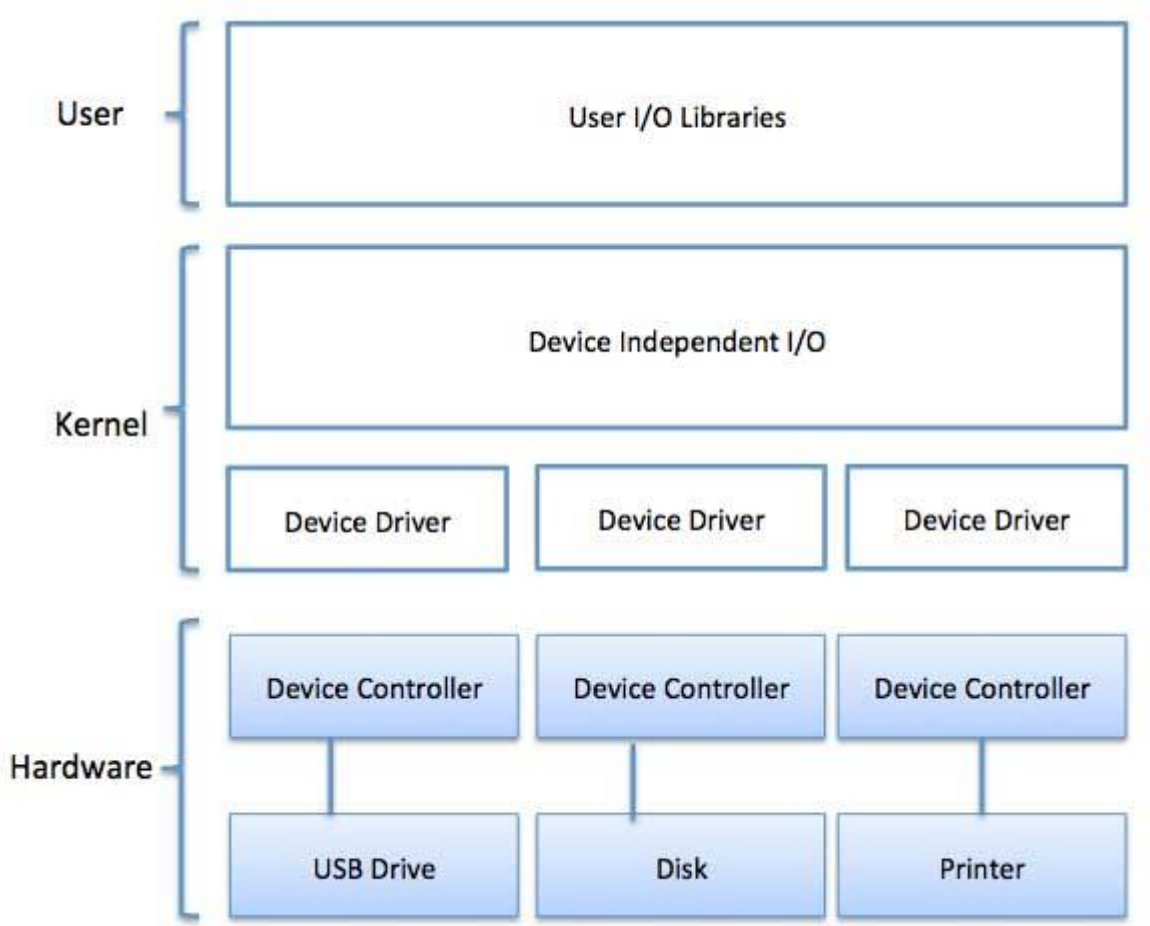
A device controller puts an interrupt signal on the bus when it needs CPU's attention when CPU receives an interrupt, It saves its current state and invokes the appropriate interrupt handler using the interrupt vector (addresses of OS routines to handle various events). When the interrupting device has been dealt with, the CPU continues with its original task as if it had never been interrupted.

PRINCIPLE OF I/O Software:

I/O software is often organized in the following layers –

- **User Level Libraries** – This provides simple interface to the user program to perform input and output. For example, **stdio** is a library provided by C and C++ programming languages.
- **Kernel Level Modules** – This provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
- **Hardware** – This layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.

A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on a floppy disk, on a hard disk, or on a CD-ROM, without having to modify the program for each different device.



Device Drivers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices. Device drivers encapsulate device-dependent code and implement a standard interface in such a way that code contains device-specific register reads/writes. Device driver, is generally written by the device's manufacturer and delivered along with the device on a CD-ROM.

A device driver performs the following jobs –

- To accept request from the device independent software above to it.
- Interact with the device controller to take and give I/O and perform required error handling
- Making sure that the request is executed successfully

How a device driver handles a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the

request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.

Interrupt handlers

An interrupt handler, also known as an interrupt service routine or ISR, is a piece of software or more specifically a callback function in an operating system or more specifically in a device driver, whose execution is triggered by the reception of an interrupt.

When the interrupt happens, the interrupt procedure does whatever it has to in order to handle the interrupt, updates data structures and wakes up process that was waiting for an interrupt to happen.

The interrupt mechanism accepts an address — a number that selects a specific interrupt handling routine/function from a small set. In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Device-Independent I/O Software

The basic function of the device-independent software is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software. Though it is difficult to write completely device independent software but we can write some modules which are common among all the devices. Following is a list of functions of device-independent I/O Software –

- Uniform interfacing for device drivers
- Device naming - Mnemonic names mapped to Major and Minor device numbers
- Device protection
- Providing a device-independent block size
- Buffering because data coming off a device cannot be stored in final destination.
- Storage allocation on block devices
- Allocation and releasing dedicated devices
- Error Reporting

User-Space I/O Software

These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers. Most of the user-level I/O software consists of library procedures with some exception like spooling system which is a way of dealing with dedicated I/O devices in a multiprogramming system.

I/O Libraries (e.g., stdio) are in user-space to provide an interface to the OS resident device-independent I/O SW. For example putchar(), getchar(), printf() and scanf() are example of user level I/O library stdio available in C programming.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** – Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
- **Buffering** – Kernel I/O Subsystem maintains a memory area known as **buffer** that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.
- **Caching** – Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- **Spooling and Device Reservation** – A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
- **Error Handling** – An operating system that uses protected memory can guard against many kinds of hardware and application errors.

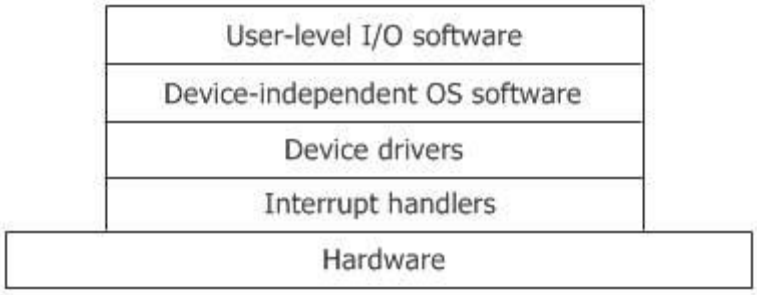
I/O SOFTWARE LAYER:

Basically, input/output software organized in the following four layers:

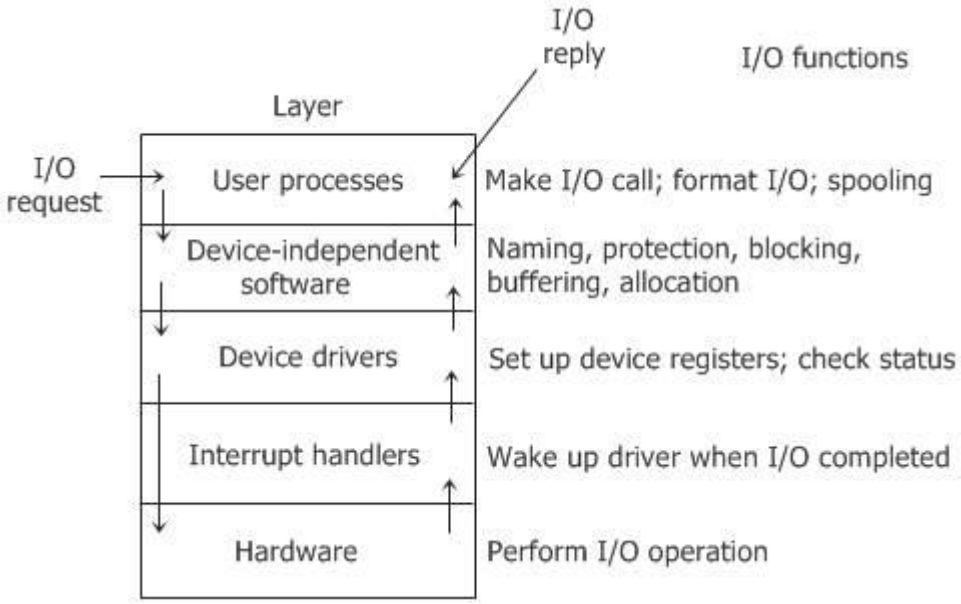
- Interrupt handlers
- Device drivers
- Device-independent input/output software
- User-space input/output software

In every input/output software, each of the above given four layer has a well-defined function to perform and a well-defined interface to the adjacent layers.

The figure given below shows all the layers along with hardware of the input/output software system.



Here is another figure shows all the layers of the input/output software system along with their principal functions.



Now let's describe briefly, all the four input/output software layers that are listed above.

Interrupt Handlers

Whenever the interrupt occurs, then the interrupt procedure does whatever it has to in order to handle the interrupt.

Device Drivers

Basically, device drivers is a device-specific code just for controlling the input/output device that are attached to the computer system.

Device-Independent Input/Output Software

In some of the input/output software is device specific, and other parts of that input/output software are device-independent.

The exact boundary between the device-independent software and drivers is device dependent, just because of that some functions that could be done in a device-independent way sometime be done in the drivers, for efficiency or any other reasons.

Here are the list of some functions that are done in the device-independent software:

- Uniform interfacing for device drivers
- Buffering
- Error reporting
- Allocating and releasing dedicated devices
- Providing a device-independent block size

User-Space Input/Output Software

Generally most of the input/output software is within the operating system (OS), and some small part of that input/output software consists of libraries that are linked with the user programs and even whole programs running outside the kernel.

DISKS:

There are a variety of types, that disk comes. Magnetic disk is the most common among those variety.

Various types of optical disks such as CD-ROMs, DVDs, CD-Recordable, are also important for distribution of programs, data, and videos.

You will learn all about disks in operating system in detail, divided into following tutorials:

- Disks
- Disk Hardware
- Disk Formatting
- Stable Storage

This tutorial of disk hardware describes briefly about the following disk hardware:

- Magnetic disks
- RAID
- CD-ROM
- CD-Recordable

- CD-Rewritable
- DVD

Magnetic Disks

Magnetic disks are organised into cylinders where each one containing as many tracks as there are heads stacked vertically. And the tracks are again divided into sectors.

RAID

RAID stands for Redundant Array of Inexpensive Disks.

The basic idea behind Redundant Array of Inexpensive Disks is just to install a box full of disks next to the computer, typically a large server, replace the disk controller card with a RAID controller, copy the data over to the RAID and then continue the normal operation.

In other word, you can say that RAID looks like a SLED (Single Large Expensive Disk) to the OS. But RAID have better performance and reliability than SLED.

CD-ROM

In the year of 1980, major two company namely Philips and Sony, together developed the Compact Disk (CD) which rapidly replaced 33 1/3-rpm vinyl record for music.

All the compact disks are 120mm across and 1.2mm thick, with a 15mm hole in the middle of the CD.

Now, after almost 4 years, that is, in the year of 1984, these two company now realised the power and potential of using the compact disks to store the computer data, therefore they developed CD-ROM, that stands for Compact Disk-Read Only Memory.

CD-Recordable

Physically, CD-Recordable (CD-R) stars with 120mm polycarbonate blanks that are like CD-ROMs, except that they contain a 0.6mm wide groove to guide the laser for writing.

CD-Rewritable

After the people's demand for a rewritable CD-ROM, a technology was developed named CD-ReWritable (CD-RW), that uses the same size media as CD-Recordable (CD-R). But, CD-RW uses an alloy of silver, antimony, indium, and tellurium for the recording layer.

DVD

DVD stands for Digital Video Disk or Digital Versatile Disk.

DVD uses the same general design as of CD with 120mm injection-moulded polycarbonate disks containing pits and lands that are illuminated by a laser diode and read by a photodetector.

DISK FORMATTING:

Basically a hard disk consists of a stack of aluminium, alloy, or glass platters 5.25, 3.5 inch or even smaller than this in diameter. On each platter is deposited on thin magnetizable metal oxide. There is no any information whatsoever on the disk after manufacturing it. Each platter of the disk must receive a low-level format done by the software before the disk can be used.

The format consists of a series of concentric track where each tracks contains some number of sectors having short gaps between the sectors.

The figure given below shows the format of a sector:

The last level in designing/developing the disk for use is to perform high-level format of each partition.

This operation of high-level format lays down a boot block, the free storage administration, root directory, and an empty file system.

The high-level format operation also puts a code in the partition table entry telling which file system is used in the partition because many OSs multiple incompatible file systems. Therefore at this point the computer system can be booted.

Now, when the power of the computer system is turned on, then the BIOS runs initially and then reads in the master boot record and jumps to it.

This boot program then checks to see that which partition is active.

Now after checking the partition for its activeness, it reads in the boot sector from that partition and runs it.

Basically the boot sector of a computer system contains a small code that searches the root directory for a specific program. That certain program is then loaded into the memory and executed.

Stable storage:

Stable storage means a storage of the data that are stable.

It is essential for some computer applications that data never be lost/corrupted even in the face of disk and central processing unit (CPU) errors.

Basically a disk should works all the time without any errors. But unfortunately it is not achievable.

Here, achievable is that a disk subsystem that has this property; when a write is issued to the disk, then the disk is either correctly writes the data or it does nothing, leaving the existing data intact. Technically, this type of system is also called as stable storage.

Basically stable storage uses a pair of identical disks with corresponding blocks working together to form a error-free block.

The corresponding blocks on both the drives are same in absense of errors. Either one can read to get same result.

The three operations which are described in the table given below are defined to achieve the goal.

Operation	Description
Stable Writes	It consists of first writing the block on the drive 1 and then reading it back just to verify that whether or not it was written correctly. If not, then the write and re-read are done again up to n times until they work.
Stable Reads	It first reads the block from the drive 1. Now in case, if this yields an incorrect ECC, then the read is tried again up to n times. And in case if all give bad ECCs, then the corresponding block is read from the drive 2. A successful stable write leaves two good copies of block behind and the probability of same block spontaneously going bad on both the drives in a reasonable time interval is negligible. Therefore, a stable read always succeeds.
Stable Recovery	Whenever a crash occurs, then a recovery program/code scans both the disks comparing corresponding blocks. Now, if blocks pair, both good and same, then nothing is done. And when one of that block pair has an ECC error, then the bad block is overwritten with the corresponding good block.

Power Management :

For your information, the first general-purpose electronic computer system named the ENIAC consumed about 140 thousands watt of power as it has 18 thousand vacuum tubes. Therefore due to consuming those high amount of watt power, it produced non-trivial electricity bill. After that, there was an invention of transistor, and the power usage dropped in large amount.

Now, in today's computer world, the power management is a very useful task for many reasons. OS is playing a big role in power management.

Generally a desktop personal computer has a 100 or 200 watt power supply.

Let's suppose that if 100 million personal computers each having 200 watt power supply, are Turned On at the same instance of time in the whole world, then it means that together they use 20 thousands mega watts of electricity. And this is the total output of 20 nuclear power plants of average-size.

Now if the power supply could be cut in half, then we could get rid of almost 10 average-size nuclear power plants.

The batter-powered computer systems such as laptops, notebooks, palmtops etc. are another place where power is a big issue.

Therefore power management is also essential in these types of computer systems.

The main problem in these type of battery-powered computer systems is that the batteries can't hold the enough charge to last very long a few more hours.

Therefore everyone's main aim in making those type of battery-powered computer systems that uses less energy just to make the existing batteries last longer is high.

The OS plays a major role here in saving the battery to use the same for more time.

Generally, there are the following two things to reduce the energy consumption:

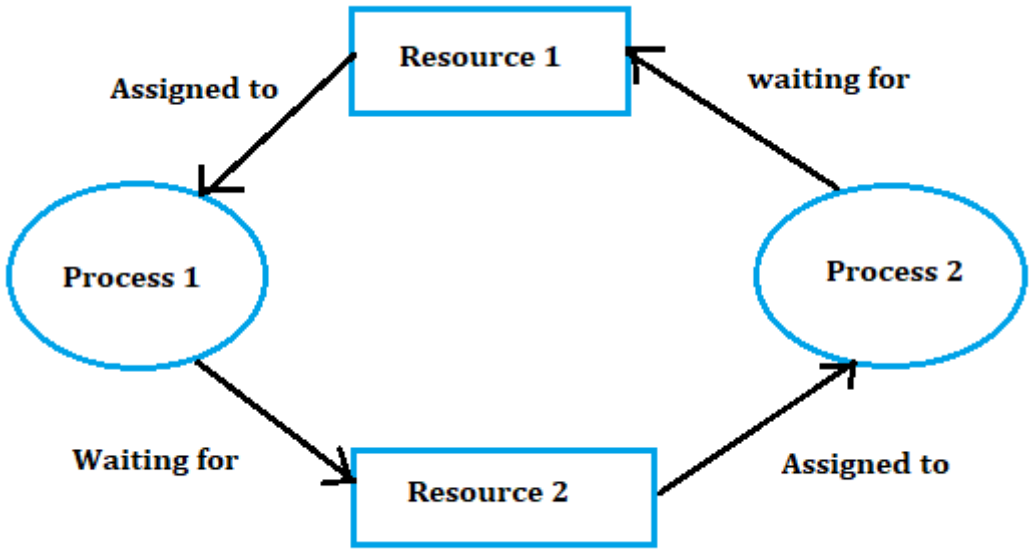
- For OS to Turn Off all the parts of the computer system that are not in use or when they are not in use. The turned off device/part uses little or no energy.

- For application programs to use less energy, by degrading the quality of the user experience to increase the battery time.

introduction to deadlocks,

A process in operating systems uses different resources and uses resources in the following way.
1) Requests a resource
2) Use the resource
2) Releases the resource

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



Deadlock can arise if the **following four conditions hold simultaneously (Necessary Conditions)**

Mutual Exclusion
: One or more than one resource are non-shareable (Only one process can #use at a time)
Hold and Wait*:* A process is holding at least one resource and waiting for resources.
No Preemption*:* A resource cannot be taken from a process unless the process releases

theresource.
Circular Wait*:* A set of processes are waiting for each other in circular form.

Methods for handling deadlock

There are three ways to handle deadlock
1) Deadlock prevention or avoidance: The idea is to not let the system into a deadlock state.
One can zoom into each category individually, Prevention is done by negating one of above mentioned necessary conditions for deadlock.
Avoidance is kind of futuristic in nature. By using strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources which process will need are known to us prior to execution of the process. We use Banker's algorithm (Which is in-turn a gift from Dijkstra) in order to avoid deadlock.
2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occurred.
3) Ignore the problem altogether: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

Ostrich Algorithm

The ostrich algorithm means that the deadlock is simply ignored and it is assumed that it will never occur. This is done because in some systems the cost of handling the deadlock is much higher than simply ignoring it as it occurs very rarely. So, it is simply assumed that the deadlock will never occur and the system is rebooted if it occurs by any chance.

deadlock detection and recovery,

Deadlock detection, deadlock prevention and deadlock avoidance are the main methods for handling deadlocks. Details about these are given as follows −

Deadlock Detection

Deadlock can be detected by the resource scheduler as it keeps track of all the resources that are allocated to different processes. After a deadlock is detected, it can be handed using the given methods −

- All the processes that are involved in the deadlock are terminated. This approach is not that useful as all the progress made by the processes is destroyed.
- Resources can be preempted from some processes and given to others until the deadlock situation is resolved.

Deadlock Prevention

It is important to prevent a deadlock before it can occur. So, the system checks each transaction before it is executed to make sure it does not lead to deadlock. If there is even a slight possibility that a transaction may lead to deadlock, it is never allowed to execute.

Some deadlock prevention schemes that use timestamps in order to make sure that a deadlock does not occur are given as follows −

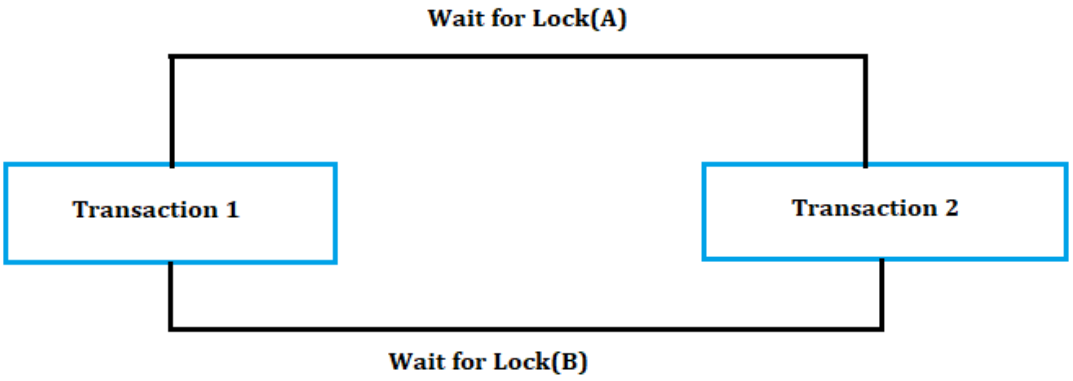
- **Wait - Die Scheme**
- In the wait - die scheme, if a transaction T1 requests for a resource that is held by transaction T2, one of the following two scenarios may occur –
    - TS(T1) < TS(T2) - If T1 is older than T2 i.e T1 came in the system earlier than T2, then it is allowed to wait for the resource which will be free when T2 has completed its execution.
    - TS(T1) > TS(T2) - If T1 is younger than T2 i.e T1 came in the system after T2, then T1 is killed. It is restarted later with the same timestamp.
- **Wound - Wait Scheme**
- In the wound - wait scheme, if a transaction T1 requests for a resource that is held by transaction T2, one of the following two possibilities may occur –
    - TS(T1) < TS(T2) - If T1 is older than T2 i.e T1 came in the system earlier than T2, then it is allowed to roll back T2 or wound T2. Then T1 takes the resource and completes its execution. T2 is later restarted with the same timestamp.
    - TS(T1) > TS(T2) - If T1 is younger than T2 i.e T1 came in the system after T2, then it is allowed to wait for the resource which will be free when T2 has completed its execution.

Deadlock Avoidance

It is better to avoid a deadlock rather than take measures after the deadlock has occurred. The wait for graph can be used for deadlock avoidance. This is however only useful for smaller databases as it can get quite complex in larger databases.

Wait for graph

The wait for graph shows the relationship between the resources and transactions. If a transaction requests a resource or if it already holds a resource, it is visible as an edge on the wait for graph. If the wait for graph contains a cycle, then there may be a deadlock in the system, otherwise not.



UNIT:4

Virtualization and Cloud: History, requirements for virtualization, type 1 and 2 hypervisors, techniques for efficient virtualization, hypervisor microkernels, memory virtualization, I/O virtualization, Virtual appliances, virtual machines on multicore CPUs, Clouds.

Multiple Processor Systems Multiprocessors, multicomputers, distributed systems.

Advantage of Virtualization:

1. A failure in one virtual machine does not bring down any others. On a virtualized system, different servers can run on different virtual machines, thus maintaining the partial-failure model that a multicomputer has, but at a lower cost and with easier maintainability. Moreover, we can now run multiple different operating systems on the same hardware, benefit from virtual machine isolation in the face of attacks, and enjoy other good stuff.

2. Other is that having fewer physical machines saves money on hardware and electricity and takes up less rack space. For a company such as Amazon or Microsoft, which may have hundreds of thousands of servers doing a huge variety of different tasks at each data center, reducing the physical demands on their data centers represents a huge cost savings. In fact, server companies frequently locate their data centers in the middle of nowhere—just to be close to, say, hydroelectric dams (and cheap energy).

3. Virtualization also helps in trying out new ideas. Typically, in large companies, individual departments or groups think of an interesting idea and then go out and buy a server to implement it. If the idea catches on and hundreds or thousands of servers are needed, the corporate data center expands. It is often hard to move the software to existing machines because each application often needs a different version of the operating system, its own libraries, configuration files, and more. With virtual machines, each application can take its own environment with it.

4. Another advantage of virtual machines is that check pointing and migrating virtual machines (e.g., for load balancing across multiple servers) is much easier than migrating processes running on a normal operating system. In the latter case, a fair amount of critical state information about every process is kept in operating system tables, including information relating to open files, alarms, signal handlers, and more.

5. Run legacy applications on operating systems (or operating system versions) no longer supported or which do not work on current hardware. These can run at the same time and on the same hardware as current applications.

6. Yet another important advantage of virtual machines is for software development

Requirements For Virtualization:-

Hypervisors should score well in there dimensions:

1. Safety: hypervisor should have full control of virtualized resources.
2. Fidelity: behaviour of a program on a virtual machine should be identical to same program running on bare hardware.
3. Efficiency :much of code in virtual machine should run without intervention by hypervisor.

Type 1 Hypervisor:

- This is also known as Bare Metal or Embedded or Native Hypervisor.
- It works directly on the hardware of the host and can monitor operating systems that run above the hypervisor.
- It is completely independent from the Operating System.
- The hypervisor is small as its main task is sharing and managing hardware resources between different operating systems.
- A major advantage is that any problems in one virtual machine or guest operating system do not affect the other guest operating systems running on the hypervisor.

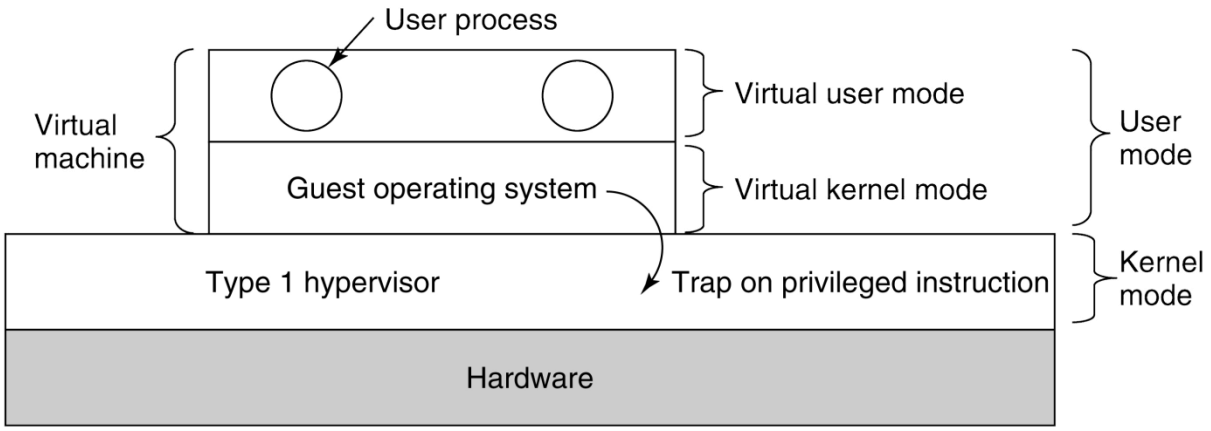
Examples:VMware ESXi Server,Microsoft Hyper-V,Citrix/Xen Server

Type 2 Hypervisor

- This is also known as Hosted Hypervisor.
- In this case, the hypervisor is installed on an operating system and then supports other operating systems above it.
- It is completely dependent on host Operating System for its operations
- While having a base operating system allows better specification of policies, any problems in the base operating system a ffects the entire system as well even if the hypervisor running above the base OS is secure.

Examples: VMware Workstation,Microsoft Virtual PC,Oracle Virtual Box.

Techniques For Efficient Virtualization:

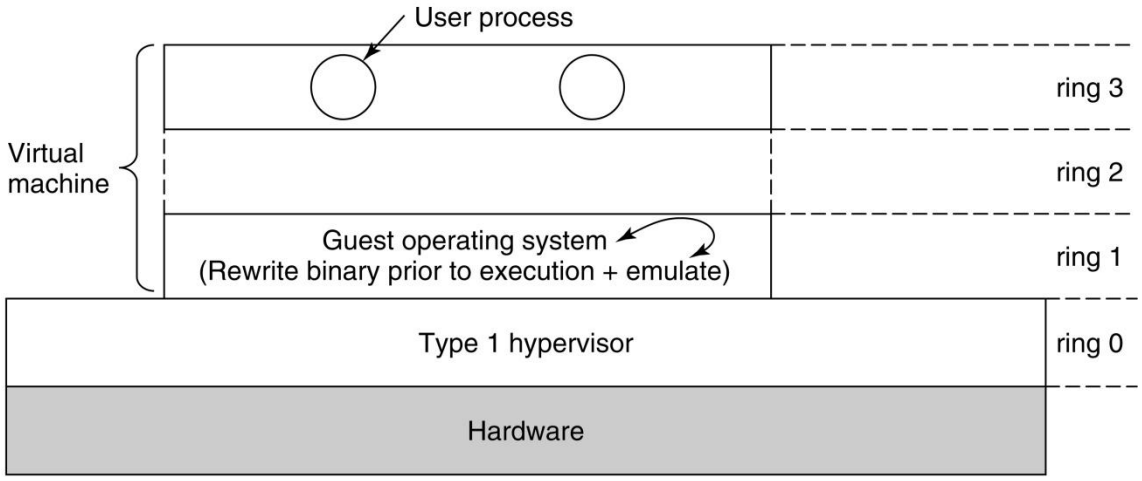


Scenario: a type 1 hypervisor is running a guest OS that thinks it is the kernel, but really is in user mode. The guest runs in virtual kernel mode. Its apps think they are in user mode (and really are).

What happens when the guest executes a privileged instruction? Normally, this would cause the program to crash. But with VT, the instruction traps to the hypervisor, and the hypervisor determines if the guest OS (or at least its kernel) called it -- OK! -- or a guest application did -- not OK!

Virtualizing the Unvirtualizable:

So how was this handled before VT was available? Using techniques like binary translation and protection rings.



Binary translator rewriting the guest OS, which runs in ring 1, while the hypervisor runs in ring 0. The user programs remain in ring 3.

Sensitive instructions are eliminated by re-writing the guest's kernel code, one **basic block** at a time. A basic block is a run of consecutive instructions, where the instruction pointer advances a single instruction

at a time. Sensitive instructions are replaced with calls to hypervisor procedures. Translated blocks are cached, so that they do not need to be translated again.

Type II hypervisors are little bit different, in that they rely upon the host's device drivers. They are like teenagers hosting a party while their parents are away: they can move everything, so long as it is all back in place when the parents return. This restoration is called a **world switch**.

The Cost of Virtualization:

Do CPUs with VT outperform software-based approaches?
Not necessarily! VT generates lots of traps, and traps are expensive. Sometimes software beats hardware: so sometimes hypervisors translate anyway, even if there is hardware support.

An example where translation is cheaper:
Turning off interrupts may be very expensive, if there are deep caches and out-of-order execution.
But a hypervisor need not *really* turn off interrupts: it only needs to make it look to the guest OS *as if* they are off. So it can just keep a binary flag per guest OS: much cheaper!

I/O virtualization: :

I/O virtualization technology allows a single physical adapter to be visualized as multiple virtual network interface cards (vNICs) and virtual host bus adapters(vHBAs). Both vNICs and vHBAs function like any other NIC and HBA card installed on the server. They work with the existing operating systems, Hypervisor and applications. The identity of the virtual cards are not disclosed to the networking resources such as LAN and SAN and continue to appear as normal cards.

I/O virtualization addresses various problems related to performance bottlenecks. Virtual I/O machines that consists of Quality of service (QoS) controls can assign I/O bandwidth to a particluar virtual device thereby allowing critical applications to deliver better performance.

Some of the major advantages of I/O Virtualization are listed below:

- Flexibilty: Since I/O virtualization involves abstracting the upper layer protocols from the underlying physical connections, it offers greater flexibilty, utilization and faster provisioning in comparison with normal NIC and HBA cards.
- cost minimization: I/O virtualization methodology involoves using fewer cables, cards and switch ports without compromising on network I/O performance.
- Increased density: I/O virtualization increases the practical density of I/O by allowing more connections to exist in a given space.
- Minimizing cables: The I/O virtualization helps to reduce the multiple cables needed to connect servers to storage and network.

Virtual appliance:

A virtual appliance is a software application residing and operating in a preconfigured virtual environment or platform. Virtual appliances are accessed remotely by users and do not require locally-installed hardware.

Applications of all sizes and complexity can be hosted through remote infrastructures. Virtual machines mirror typically installed computer OSs, but do not contain software applications. In other words, a virtual appliance is essentially a software appliance that is installed on a virtual machine.

Virtual appliances have several benefits, particularly ease of deployment. Users are not responsible for managing hardware and software compatibility or OS considerations such as integration and isolation in the event of a system crash. Virtual appliances play a significant role in cloud computing's software as a service (SaaS) model where remote software access is delivered through a Web browser.

Virtual machine in multicore CPU:

A virtual machine (VM) is an image file managed by the hypervisor that exhibits the behavior of a separate computer, capable of performing tasks such as running applications and programs like a separate computer.

In other words, a VM is a software application that performs most functions of a physical computer, actually behaving as a separate computer system.

A virtual machine, usually known as a guest, is created within another computing environment referred as a "host." Multiple virtual machines can exist within a single host at one time.

Virtual Machine (VM)

Virtual machines are becoming more common with the evolution of virtualization technology. Virtual machines are often created to perform certain tasks that are different than tasks performed in a host environment.

They are also widely implemented as a sandboxed environment that are separated from the rest of the network. For example, they can be used for testing purposes, especially to perform risky tasks such as running malicious software, testing operating systems, and accessing malware-infected data.

VMs are also used in production and as back-ups.

Virtual machines are implemented by software emulation methods or hardware virtualization techniques. A lightweight software known as hypervisor allocates the computing resources (RAM, CPU power, memory, storage, etc.) of the server or host to each VM, keeping all of them separate to avoid interference.

The computer's operating system and applications are separated from its hardware so that each new virtual machine can access the physical resources of the original server, which are managed by the hypervisor.

The VM has virtual hardware resources that map to the physical hardware on the server (host). This allows for load balancing of resources across VMs on a single host.

Depending on their use and level of correspondence to any physical computer, virtual machines can be divided into two categories:

System Virtual Machines

A system platform that supports the sharing of the host computer's physical resources between multiple virtual machines, each running with its own copy of the operating system.

The virtualization technique is provided by the hypervisor, which can run either on bare hardware or on top of an operating system.

Process Virtual Machine

Also known as application VM, a process virtual machine is designed to provide a platform-independent programming environment that supports a single process. It is created when the process is started, and destroyed upon exit.

A process VM is used to mask the information of the underlying hardware or operating system, and allows program execution to take place in the same way on any given platform

Pros and Cons of Virtual Machines

Some of the advantages of a virtual machine include:

- Allows multiple operating system environments on a single physical computer without any intervention.
- Virtual machines are widely available and are easy to manage and maintain.
- Offers application provisioning and disaster recovery options.
- A VM can be created or replicated very quickly by cloning it with an OS already installed, rather than installing a new OS on a physical server.
- VMs are offer high availability since they can be moved from one server to another for maintenance purposes, even whilst running.

Some of the drawbacks of virtual machines include:

- They are not as efficient as a physical computer because the hardware resources are distributed in an indirect way.

- Multiple VMs running on a single physical machine can deliver unstable performance.

What is cloud and its essential characteristics of cloud.

The key idea of a cloud is straightforward: outsource your computation or storage needs to a well-managed data center-run by a company specializing in this and staffed by experts in the area. Five essential characteristics:

1. On-demand self-service. Users should be able to provision resources automatically, without requiring human interaction.

2. Broad network access. All these resources should be available over the network via standard mechanisms so that heterogeneous devices can make use of them.

3. Resource pooling. The computing resource owned by the provider should be pooled to serve multiple users and with the ability to assign and reassign resources dynamically. The users generally do not even know the exact location of ‘‘their’’ resources or even which country they are located in.

4. Rapid elasticity. It should be possible to acquire and release resources elastically, perhaps even automatically, to scale immediately with the users' demands.

5. Measured service. The cloud provider meters the resources used in a way that matches the type of service agreed upon

Multiprocessor Operating System Types:

1.Each CPU Has Its Own Operating System

1.The simplest possible way to organize a multiprocessor operating system is to statically divide memory into as many partitions as there are CPUs and give each CPU its own private memory and its own private copy of the operating system.

2. In effect, the n CPUs then operate as n independent computers. One obvious optimization is to allow all the CPUs to share the operating system code and make private copies of only the operating system data structures, as shown in Fig. 8-7.

3. This scheme is still better than having n separate computers since it allows all the machines to share a set of disks and other I/O devices, and it also allows the memory to be shared flexibly.

4. For example, even with static memory allocation, one CPU can be given an extra-large portion of the memory so it can handle large programs efficiently.

5. Second, since each operating system has its own tables, it also has its own set of processes that it schedules by itself. There is no sharing of processes. If a user logs into

CPU 1, all of his processes run on CPU 1. As a consequence, it can happen that CPU 1 is idle while CPU 2 is loaded with work.

6. Third, there is no sharing of physical pages. It can happen that CPU 1 has pages to spare while CPU 2 is paging continuously. There is no way for CPU 2 to borrow some pages from CPU 1 since the memory allocation is fixed.

7. Fourth, and worst, if the operating system maintains a buffer cache of recently used disk blocks, each operating system does this independently of the other ones. Thus it can happen that a certain disk block is present and dirty in multiple buffer caches at the same time.leading in inconsistent results.

2.Master-Slave Microprocessor:

1.A second model is shown in Fig. 8-8. Here, one copy of the operating system and its tables is present on CPU 1 and not on any of the others.

2. All system calls are redirected to CPU 1 for processing there. CPU 1 may also run user processes if there is CPU time left over. This model is called master-slave since CPU 1 is the master and all the others are slaves.

3.The master-slave model solves most of the problems of the first model. There is a single data structure (e.g., one list or a set of prioritized lists) that keeps track of ready processes.

4. When a CPU goes idle, it asks the operating system on CPU 1 for a process to run and is assigned one.

5. Thus it can never happen that one CPU is idle while another is overloaded. Similarly, pages can be allocated among all the processes dynamically and there is only one buffer cache, so inconsistencies never occur.

3. Symmetric Multiprocessors

1. Our third model SMP,eliminates this asymmetry.There is a one of the copy of operating system In memory, but any CPU can run it.

2.When a system call is made, the CPU on which the system call was made traps to the kernel and processes the system call. The SMP model is illustrated in Fig. 8-9.

3. This model balances processes and memory dynamically, since there is only one set of operating system tables.

4. It also eliminates the master CPU bottleneck, since there is no master, but it introduces its own problems.

5. In particular, if two or more CPUs are running operating system code at the same time, disaster may well result. Imagine two CPUs simultaneously picking the same process to run or claiming the same free memory page.

6. The simplest way around these problems is to associate a mutex (i.e., lock) with the operating system, making the whole system one big critical region.

7.When a CPU wants to run operating system code, it must first acquire the mutex. If the mutex is locked, it just waits.

8. In this way, any CPU can run the operating system, but only one at a time. This approachis somethings called a big kernel lock.

Advantages of Multiprocessor Systems

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

More reliable Systems

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

Enhanced Throughput

If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.

More Economic Systems

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

Disadvantages of Multiprocessor Systems

There are some disadvantages as well to multiprocessor systems. Some of these are:

Increased Expense

Even though multiprocessor systems are cheaper in the long run than using multiple computer systems, still they are quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.

Complicated Operating System Required

There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart

resources to processes.than in single processor systems. Hence, a more complex and complicated operating system is required in multiprocessor systems.

Large Main Memory Required

All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

Multiprocessor System Interconnects

Parallel processing needs the use of efficient system interconnects for fast communication among the Input/Output and peripheral devices, multiprocessors and shared memory.

Hierarchical Bus Systems

A hierarchical bus system consists of a hierarchy of buses connecting various systems and sub-systems/components in a computer. Each bus is made up of a number of signal, control, and power lines. Different buses like local buses, backplane buses and I/O buses are used to perform different interconnection functions.

Local buses are the buses implemented on the printed-circuit boards. A backplane bus is a printed circuit on which many connectors are used to plug in functional boards. Buses which connect input/output devices to a computer system are known as I/O buses.

Crossbar switch and Multiport Memory

Switched networks give dynamic interconnections among the inputs and outputs. Small or medium size systems mostly use crossbar networks. Multistage networks can be expanded to the larger systems, if the increased latency problem can be solved.

Both crossbar switch and multiport memory organization is a single-stage network. Though a single stage network is cheaper to build, but multiple passes may be needed to establish certain connections. A multistage network has more than one stage of switch boxes. These networks should be able to connect any input to any output.

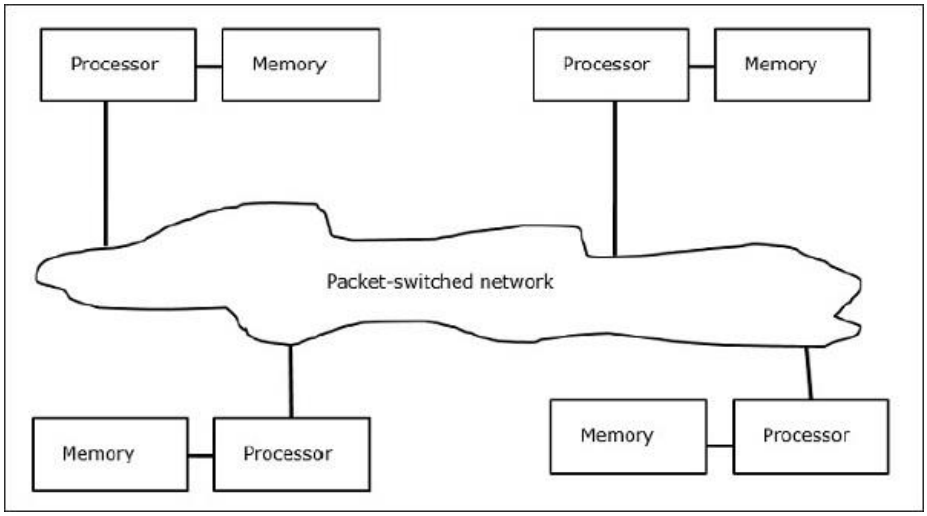
Multistage and Combining Networks

Multistage networks or multistage interconnection networks are a class of high-speed computer networks which is mainly composed of processing elements on one end of the network and memory elements on the other end, connected by switching elements.

These networks are applied to build larger multiprocessor systems. This includes Omega Network, Butterfly Network and many more.

Multicomputers:

Multicomputers are distributed memory MIMD architectures. The following diagram shows a conceptual model of a multicomputer –



Multicomputers are message-passing machines which apply packet switching method to exchange data. Here, each processor has a private memory, but no global address space as a processor can access only its own local memory. So, communication is not transparent: here programmers have to explicitly put communication primitives in their code.

Having no globally accessible memory is a drawback of multicomputers. This can be solved by using the following two schemes –

- Virtual Shared Memory (VSM)
- Shared Virtual Memory (SVM)

In these schemes, the application programmer assumes a big shared memory which is globally addressable. If required, the memory references made by applications are translated into the message-passing paradigm.

Virtual Shared Memory (VSM)

VSM is a hardware implementation. So, the virtual memory system of the Operating System is transparently implemented on top of VSM. So, the operating system thinks it is running on a machine with a shared memory.

Shared Virtual Memory (SVM)

SVM is a software implementation at the Operating System level with hardware support from the Memory Management Unit (MMU) of the processor. Here, the unit of sharing is Operating System memory pages.

If a processor addresses a particular memory location, the MMU determines whether the memory page associated with the memory access is in the local memory or not. If the page is not in the memory, in a normal computer system it is swapped in from the disk by the Operating System. But, in SVM, the Operating System fetches the page from the remote node which owns that particular page.

Three Generations of Multicomputers

In this section, we will discuss three generations of multicomputers.

Design Choices in the Past

While selecting a processor technology, a multicomputer designer chooses low-cost medium grain processors as building blocks. Majority of parallel computers are built with standard off-the-shelf microprocessors. Distributed memory was chosen for multi-computers rather than using shared memory, which would limit the scalability. Each processor has its own local memory unit.

For interconnection scheme, multicomputers have message passing, point-to-point direct networks rather than address switching networks. For control strategy, designer of multi-computers choose the asynchronous MIMD, MPMD, and SMPD operations. Caltech's Cosmic Cube (Seitz, 1983) is the first of the first generation multi-computers.

Present and Future Development

The next generation computers evolved from medium to fine grain multicomputers using a globally shared virtual memory. Second generation multi-computers are still in use at present. But using better processor like i386, i860, etc. second generation computers have developed a lot.

Third generation computers are the next generation computers where VLSI implemented nodes will be used. Each node may have a 14-MIPS processor, 20-Mbytes/s routing channels and 16 Kbytes of RAM integrated on a single chip.

The Intel Paragon System

Previously, homogeneous nodes were used to make hypercube multicomputers, as all the functions were given to the host. So, this limited the I/O bandwidth. Thus to solve large-scale problems efficiently or with high throughput, these computers could not be used.The Intel Paragon System was designed to overcome this difficulty. It turned the multicomputer into an application server with multiuser access in a network environment.

Message Passing Mechanisms

Message passing mechanisms in a multicomputer network needs special hardware and software support. In this section, we will discuss some schemes.

Message-Routing Schemes

In multicomputer with store and forward routing scheme, packets are the smallest unit of information transmission. In wormhole–routed networks, packets are further divided into flits. Packet length is determined by the routing scheme and network implementation, whereas the flit length is affected by the network size.

In **Store and forward routing**, packets are the basic unit of information transmission. In this case, each node uses a packet buffer. A packet is transmitted from a source node to a destination node through a sequence of intermediate nodes. Latency is directly proportional to the distance between the source and the destination.

In **wormhole routing**, the transmission from the source node to the destination node is done through a sequence of routers. All the flits of the same packet are transmitted in an inseparable sequence in a pipelined fashion. In this case, only the header flit knows where the packet is going.

Deadlock and Virtual Channels

A virtual channel is a logical link between two nodes. It is formed by flit buffer in source node and receiver node, and a physical channel between them. When a physical channel is allocated for a pair, one source buffer is paired with one receiver buffer to form a virtual channel.

When all the channels are occupied by messages and none of the channel in the cycle is freed, a deadlock situation will occur. To avoid this a deadlock avoidance scheme has to be followed.

Distributed System:-

An operating system (OS) is basically a collection of software that manages computer hardware resources and provides common services for computer programs. Operating system is a crucial component of the system software in a computer system.

Distributed Operating System is one of the important type of operating system.

Multiple central processors are used by Distributed systems to serve multiple real-time applications and multiple users. Accordingly, Data processing jobs are distributed among the processors.

Processors communicate with each other through various communication lines (like high-speed buses or telephone lines). These are known as **loosely coupled systems** or distributed systems. Processors in this system may vary in size and function. They are referred as sites, nodes, computers, and so on.

Advantages

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- Failure of one site in a distributed system doesn't affect the others, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

UNIT :5

Case Study on LINUX and ANDROID: History of Unix and Linux, Linux Overview, Processes in Linux, Memory management in Linux, I/O in Linux, Linux file system, security in Linux. Android

Case Study on Windows: History of windows through Windows 10, programming windows, system structure, processes and threads in windows, memory management, caching in windows, I/O in windows, Windows NT file system, Windows power management, Security in windows.

History of Unix and Linux:-

Unix History:

Unix is a computer operating system developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thosmpos, Dennis Ritchie, Brian Kernighan, Douglas McIlroy and Joe Ossanna.

In 1965 AT&T bell Labs,MIT and General Electric developed operating system called MULTICS (Multiplexed Information and Computing System).Bell Labs dropped out of the Multics project in April 1969.AT & T employee Ken Thompson and Dennis Ritchie lead an OS team and continued to work on the same project and developed UNICS (Uniplexed Information and Computing System). Later on it was named UNIX.

In 1973 Unix was written in C .soon more than 600 companies started using UNIX. AT&T made Unix available to universities and commercial firms, as well as the United States government under licenses. The licenses included all the source code including the machine-depended parts of the kernel. soon other companies began to offer commercial versions of UNIX.

Linux History:

Linux is a free Unix-like operating system kernel created by Linus Torvalds and released under the GNU General Public License.Linux is an open-source version of the UNIX operating system. Actually we should call it GNU Linux rather than just Linux.

Linux is the kernel, one of the essential major components of the system. But as this is the first article for those who want to know more about Linux,we will time being call it just Linux. Linux is a generic term referring to Unix-like graphical user interface (GUI) based computer operating systems based on the Linux kernel.Linux is freely-distributable implementation of UNIX that runs on a number of hardware platforms, including Intel and Motorola microprocessors.

Linux has a reputation as a very efficient and fast-performing system. Linux was invented by Linus Torvalds in 1991,when Torvalds was a student at the University of Helsinki in Finland. Linux derived Linux from MINIX, a non-free Unix -like system. MINIX was first released in 1987, with its complete source code made available to universities for study in courses and research. It has been free and open source software since it was re-licensed under the BSD license in April 2000.

Linux Overview:-

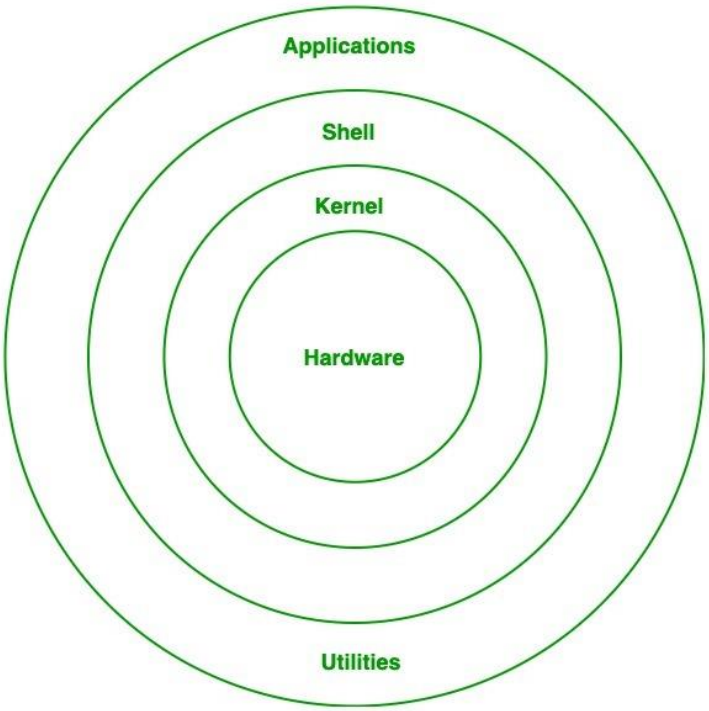
Linux Distribution

Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software. And you can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers, etc. Around **600 + Linux Distributions** are available and some of the popular Linux distributions are:

- MX Linux
- Manjaro
- Linux Mint
- elementary
- Ubuntu
- Debian
- Solus
- Fedora
- openSUSE
- Deepin

Architecture of Linux

Linux architecture has the following components:



1. **Kernel:** Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes. Different types of the kernel are:
   - Monolithic Kernel
   - Hybrid kernels
   - Exo kernels
   - Micro kernels
2. **System Library: Is**the special types of functions that are used to implement the functionality of the operating system.
3. **Shell:** It is an interface to the kernel which hides the complexity of the kernel's functions from the users. It takes commands from the user and executes the kernel's functions.
4. **Hardware Layer:** This layer consists all peripheral devices like RAM/ HDD/ CPU etc.
5. **System Utility:** It provides the functionalities of an operating system to the user.

Advantages of Linux

- The main advantage of Linux, is it is an open-source operating system. This means the source code is easily available for everyone and you are allowed to contribute, modify and distribute the code to anyone without any permissions.
- In terms of security, Linux is more secure than any other operating system. It does not mean that Linux is 100 percent secure it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software.
- The software updates in Linux are easy and frequent.

- Various Linux distributions are available so that you can use them according to your requirements or according to your taste.
- Linux is freely available to use on the internet.
- It has large community support.
- It provides high stability. It rarely slows down or freezes and there is no need to reboot it after a short time.
- It maintain the privacy of the user.
- The performance of the Linux system is much higher than other operating systems. It allows a large number of people to work at the same time and it handles them efficiently.
- It is network friendly.
- The flexibility of Linux is high. There is no need to install a complete Linux suit; you are allowed to install only required components.
- Linux is compatible with a large number of file formats.
- It is fast and easy to install from the web. It can also install on any hardware even on your old computer system.
- It performs all tasks properly even if it has limited space on the hard disk.

Disadvantages of Linux

- It is not very user-friendly. So, it may be confusing for beginners.
- It has small peripheral hardware drivers as compared to windows.

Processes In Linux:

A process is the basic context within which all user-requested activity is serviced within the operating system. To be compatible with other UNIX systems, Linux must use a process model similar to those of other versions of UNIX. Linux operates differently from UNIX in a few key places, however. In this section, we review the traditional UNIX process model (Section A.3.2) and introduce Linux's own threading model.

The fork() and exec() Process Model

The basic principle of UNIX process management is to separate two operations: the creation of a process and the rum1.ing of a new program. A new process is created by the fork() system call, and a new program is run after a call to exec(). These are two distinctly separate functions. A new process may be created with fork() without a new program being run-the new subprocess simply continues to execute exactly the same program that the first (parent) process was running. Equally, running a new program does not require that a new process be created first: any process may call exec() at any time. The currently running program is immediately terminated, and the new program starts executing in the context of the existing process.

This model has the advantage of great simplicity. It is not necessary to specify every detail of the environment of a new program in the system call that runs that program; the new program simply runs in its existing environment.

If a parent process wishes to modify the environment in which a new program is to be run, it can fork and then, still running the original program in a child process, make any system calls it requires to modify that child process before finally executing the new program. Under UNIX, then, a process encompasses all the information that the operating system must maintain to track the context of a single execution of a single program. Under Linux, we can break down this context into a number of specific sections. Broadly, process properties fall into three groups: the process identity, environment, and context.

Process Identity:-

A process identity consists mainly of the following items:

Process ID (PID). Each process has a Lmique identifier. The PID is used to specify the process to the operating system when an application makes a system call to signal, modify, or wait for the process. Additional identifiers associate the process with a process group (typically, a tree of processes forked by a single user command) and login session.

Credentials. Each process must have an associated user ID and one or more group IDs (user groups are discussed in Section 10.6.2) that determine the rights of a process to access system resources and files.

Personality. Process personalities are not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls. Personalities are primarily used by emulation libraries to request that system calls be compatible with certain varieties of UNIX. Most of these identifiers are under the limited control of the process itself. The process group and session identifiers can be changed if the process wants to start a new group or session. Its credentials can be changed, subject to appropriate security checks. Howeve1~ the primary PID of a process is unchangeable and uniquely identifies that process until termination.

Process Environment:

A process's environment is inherited from its parent and is composed of two null-terminated vectors: the argument vector and the enviromnent vector. The argument vector simply lists the command-line arguments used to invoke the running program; it conventionally starts with the name of the program itself. The environment vector is a list of "NAME= VALUE" pairs that associates named environment variables with arbitrary textual values. The environment is not held in kernel memory but is stored in the process's own user-mode address space as the first datum at the top of the process's stack. The argument and environment vectors are not altered when a new process is created; the new child process will inherit the environment that its parent possesses. However, a completely new environment is set up when a new program is invoked. On

calling exec (),a process must supply the environment for the new program. The kernel passes these environment variables to the next program, replacing the process's current environment. The kernel otherwise leaves the environment and command-line vectors alone-their interpretation is left entirely to the user-mode libraries and applications.

The passing of environment variables from one process to the next and the inheriting of these variables by the children of a process provide flexible ways to pass information to components of the user-mode system software. Various important environment variables have conventional meanings to related parts of the system software. For example, the TERM variable is set up to name the type of terminal com1.ected to a user's login session; many programs use this variable to determine how to perform operations on the user's display, such as moving the cursor and scrolling a region of text. Programs with multilingual support use the LANG variable to determine in which language to display system messages for programs that include multilingual support.

The environment-variable mechanism custom-tailors the operating system on a per-process basis, rather than for the system as a whole. Users can choose their own languages or select their own editors independently of one another.

Process Context

The process identity and environment properties are usually set up when a process is created and not changed until that process exits. A process may choose to change some aspects of its identity if it needs to do so, or it may alter its environment. In contrast, process context is the state of the running program at any one time; it changes constantly. Process context includes the following parts:

Scheduling context. The most important part of the process context is its scheduling context-the information that the scheduler needs to suspend and restart the process. This information includes saved copies of all the process's registers. Floating-point registers are stored separately and are restored only when needed, so that processes that do not use floating-point arithmetic do not incur the overhead of saving that state. The scheduling context also includes information about scheduling priority and about any outstanding signals waiting to be delivered to the process. A key part of the scheduling context is the process's kernel stack, a separate area of kernel memory reserved for use exclusively by kernel-mode code. Both system calls and interrupts that occur while the process is executing will use this stack.

Accounting. The kernel maintains accounting information about the resources currently being consumed by each process and the total resources consumed by the process in its entire lifetime so far.

File table. The file table is an array of pointers to kernel file structures. When making file-I/O system calls, processes refer to files by their index into this table.

File-system context. Whereas the file table lists the existing open files, the file-system context applies to requests to open new files. The current root and default directories to be used for new file searches are stored here. Signal-handler table. UNIX systems can deliver asynchronous signals to a process in response to various external events. The signal-handler table defines the routine in the process's address space to be called when a specific signal arrives.

Virtual memory context. The virtual memory context describes the full contents of a process's private address space;

I/O In Linux:

To the user, the I/O system in Linux looks much like that in any UNIX system. That is, to the extent possible, all device drivers appear as normal files. Users can open an access channel to a device in the same way they opens any other file-devices can appear as objects within the file system. The system administrator can create special files within a file system that contain references to a specific device driver, and a user opening such a file will be able to read from and write to the device referenced. By using the normal file-protection system, which determines who can access which file, the administrator can set access permissions for each device. Linux splits all devices into three classes: block devices, character devices, and network devices.

Block device: include all devices that allow random access to completely independent, fixed-sized blocks of data, including hard disks and floppy disks, CD-ROMs, and flash memory. Block devices are typically used to store file systems, but direct access to a block device is also allowed so that programs can create and repair the file system that the device contains. Applications can also access these block devices directly if they wish; for example, a database application may prefer to perform its own, fine-tuned laying out of data onto the disk, rather than using the general-purpose file system.

Character device: include most other devices, such as mice and keyboards. The fundamental difference between block and character devices is random access-block devices may be accessed randomly, while character devices are only accessed serially. For example, seeking to a certain position in a file might be supported for a DVD but makes no sense to a pointil1.g device such as a mouse.

Network device: are dealt with differently from block and character devices. Users cannot directly transfer data to network devices; instead, they must communicate indirectly by opening a connection to the kernel's networking subsystem.

Linux File System:

A Linux file system is a structured collection of files on a disk drive or a partition. A partition is a segment of memory and contains some specific data. In our machine, there can be various partitions of the memory. Generally, every partition contains a file system.

The general-purpose computer system needs to store data systematically so that we can easily access the files in less time. It stores the data on hard disks (HDD) or some equivalent storage type. There may be below reasons for maintaining the file system:

- Primarily the computer saves data to the RAM storage; it may lose the data if it gets turned off. However, there is non-volatile RAM (Flash RAM and SSD) that is available to maintain the data after the power interruption.
- Data storage is preferred on hard drives as compared to standard RAM as RAM costs more than disk space. The hard disks costs are dropping gradually comparatively the RAM.

The Linux file system contains the following sections:

- The root directory (/)
- A specific data storage format (EXT3, EXT4, BTRFS, XFS and so on)
- A partition or logical volume having a particular file system.

What is the Linux File System?

Linux file system is generally a built-in layer of a Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file.

If we have an unsupported file format in our file system, we can download software to deal with it.

Linux File System Structure

Linux file system has a hierarchal file structure as it contains a root directory and its subdirectories. All other directories can be accessed from the root directory. A partition usually has only one file system, but it may have more than one file system.

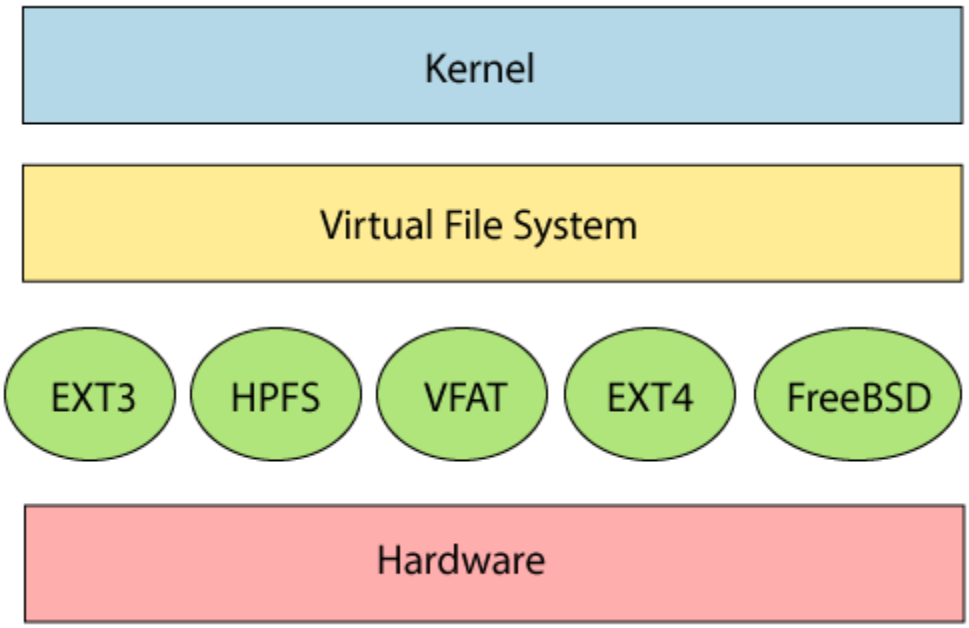
A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology. The namespace defines the naming process, length of the file name, or a subset of characters that can be used for the file name. It also defines the logical structure of files on a memory segment, such as the use of directories for organizing the specific files. Once a namespace is described, a Metadata description must be defined for that particular file.

The data structure needs to support a hierarchical directory structure; this structure is used to describe the available and used disk space for a particular block. It also has the other details about the files such as file size, date & time of creation, update, and last modified.

Also, it stores advanced information about the section of the disk, such as partitions and volumes.

The advanced data and the structures that it represents contain the information about the file system stored on the drive; it is distinct and independent of the file system metadata.

Linux file system contains two-part file system software implementation architecture. Consider the below image:



Linux File System Features

In Linux, the file system creates a tree structure. All the files are arranged as a tree and its branches. The topmost directory called the **root (/) directory**. All other directories in Linux can be accessed from the root directory.

Some key features of Linux file system are as following:

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.
- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitivity:** Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension. While working with Shell, it

creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, it symbolizes the files and folders.

- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name (e.g., .ignore). To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

Types of Linux File System

When we install the Linux operating system, Linux offers many file systems such as **Ext, Ext2, Ext3, Ext4, JFS, ReiserFS, XFS, btrfs,** and **swap**.

1. . Ext, Ext2, Ext3 and Ext4 file system

The file system Ext stands for **Extended File System**. It was primarily developed for **MINIX OS**. The Ext file system is an older version, and is no longer used due to some limitations.

Ext2 is the first Linux file system that allows managing two terabytes of data. Ext3 is developed through Ext2; it is an upgraded version of Ext2 and contains backward compatibility. The major drawback of Ext3 is that it does not support servers because this file system does not support file recovery and disk snapshot.

Ext4 file system is the faster file system among all the Ext file systems. It is a very compatible option for the SSD (solid-state drive) disks, and it is the default file system in Linux distribution.

2. JFS File System

JFS stands for **Journaled File System**, and it is developed by **IBM for AIX Unix**. It is an alternative to the Ext file system. It can also be used in place of Ext4, where stability is needed with few resources. It is a handy file system when CPU power is limited.

3. ReiserFS File System

ReiserFS is an alternative to the Ext3 file system. It has improved performance and advanced features. In the earlier time, the ReiserFS was used as the default file system in SUSE Linux, but later it has changed some policies, so SUSE returned to Ext3. This file system dynamically supports the file extension, but it has some drawbacks in performance.

4. XFS File System

XFS file system was considered as high-speed JFS, which is developed for parallel I/O processing. NASA still using this file system with its high storage server (300+ Terabyte server).

5. Btrfs File System

Btrfs stands for the **B tree file system**. It is used for fault tolerance, repair system, fun administration, extensive storage configuration, and more. It is not a good suit for the production system.

6. Swap File System

The swap file system is used for memory paging in Linux operating system during the system hibernation. A system that never goes in hibernate state is required to have swap space equal to its RAM size.

Android:

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Why Android ?

Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below –

Sr.No.	Feature & Description
1	**Beautiful UI** Android OS basic screen provides a beautiful and intuitive user interface.
2	**Connectivity** GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
3	**Storage** SQLite, a lightweight relational database, is used for data storage purposes.
4	**Media support** H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.

5	**Messaging** SMS and MMS
6	**Web browser** Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
7	**Multi-touch** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
8	**Multi-tasking** User can jump from one task to another and same time various application can run simultaneously.
9	**Resizable widgets** Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	**Multi-Language** Supports single direction and bi-directional text.
11	**GCM** Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
12	**Wi-Fi Direct** A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
13	**Android Beam** A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit.

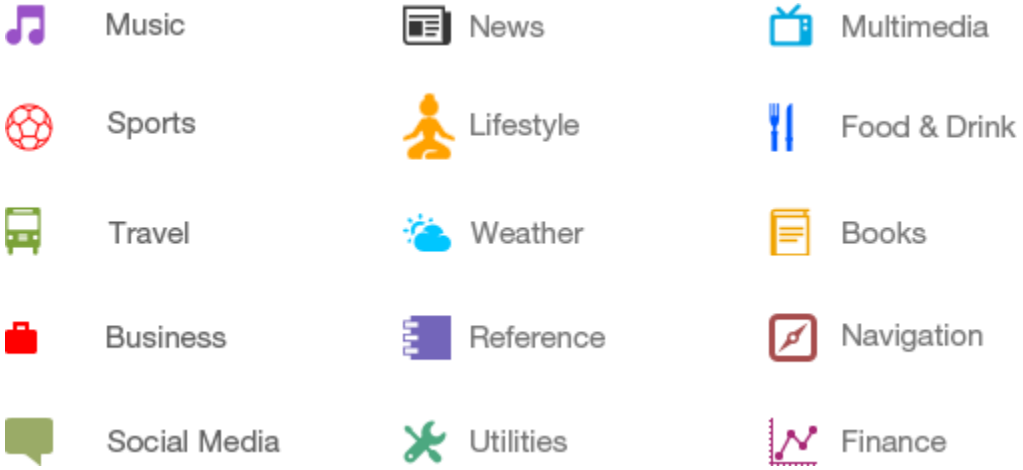
Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

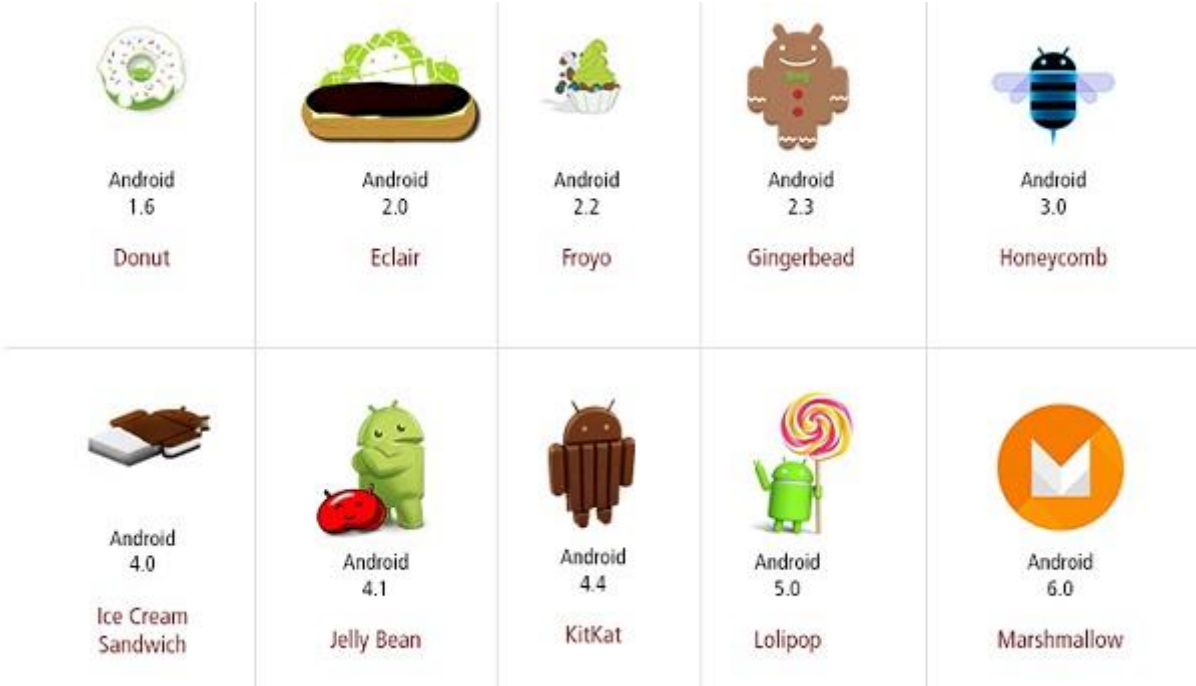
Categories of Android applications

There are many android applications in the market. The top categories are –



History of Android

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.



What is API level?

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Platform Version	**API Level**	**VERSION_CODE**	
Android 6.0	23	MARSHMALLOW	
Android 5.1	22	LOLLIPOP_MR1	
Android 5.0	21	LOLLIPOP	
Android 4.4W	20	KITKAT_WATCH	KitKat for Wearables Only
Android 4.4	19	KITKAT	
Android 4.3	18	JELLY_BEAN_MR2	

Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	
Android 4.1, 4.1.1	16	JELLY_BEAN	
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1	
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	
Android 3.2	13	HONEYCOMB_MR2	
Android 3.1.x	12	HONEYCOMB_MR1	
Android 3.0.x	11	HONEYCOMB	
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1	
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD	
Android 2.2.x	8	FROYO	
Android 2.1.x	7	ECLAIR_MR1	
Android 2.0.1	6	ECLAIR_0_1	
Android 2.0	5	ECLAIR	
Android 1.6	4	DONUT	

Android 1.5	3	CUPCAKE	
Android 1.1	2	BASE_1_1	
Android 1.0	1	BASE	

Security in Linux:

Linux's security model is closely related to typical UNIX security mechanisms.The security concerns can be classified in two groups:

1.Authentication. Making sure that nobody can access the system without first proving that she has entry rights.

2.Access control. Providing a mechanism for checking whether a user has the right to access a certain object and preventing access to objects as required.

Authentication:

Authentication in UNIX has typically been performed through the use of a publicly readable password file. A user's password is combined with a random "salt" value, and the result is encoded with a one-way transformation function and stored in the password file. The use of the one-way function means that the original password cam1ot be deduced from the password file except by trial and error. When a user presents a password to the system, the password is recombined with the salt value stored in the password file and passed through the same one-way transformation. If the result matches the contents of the password file, then the password is accepted.

Historically, UNIX implementations of this mechanism have had several problems. Passwords were often limited to eight characters, and the number of possible salt values was so low that an attacker could easily combine a dictionary of commonly used passwords with every possible salt value and have a good chance of matching one or more passwords in the password file, gaining "authorized access to any accounts compromised as a result. Extensions to the password mechanism have been introduced that keep the encrypted password secret in a file that is not publicly readable, that allow longer passwords, or that use more secure methods of encoding the password. Other authentication mechanisms have been introduced that limit the times during which a user is permitted to connect to the system. Also, mechanisms exist to distribute authentication information to all the related systems in a network. A new security mechanism has been developed by UNIX vendors to address authentication problems. The system is based on a shared library that can be used by any system component that needs to authenticate users. An implementation of this system is available under Linux.

PAM allows authentication modules to be loaded on demand as specified in a system-wide configuration file. If a new authentication mechanism is added at a later date, it can be added to the configuration file, and all system components will immediately be able to take advantage of it.

PAM modules can specify authentication methods, account restrictions, session-setup functions, and password-changing functions (so that, when users change their passwords, all the necessary authentication mechanisms can be updated at once).

Access Control

Access control under UNIX systems, including Linux, is performed through the use of unique numeric identifiers. A user identifier (UID) identifies a single user or a single set of access rights. A group identifier (GID) is an extra identifier that can be used to identify rights belonging to more than one user. Access control is applied to various objects in the system. Every file available in the system is protected by the standard access-control mechanism. In addition, other shared objects, such as shared-memory sections and semaphores, employ the same access system.

Every object in a UNIX system under user and group access control has a single UID and a single GID associated with it. User processes also have a single UID, but they may have more than one GID. If a process's UID matches the UID of an object, then the process has or to that object. If the UIDs do not match but any GID of the process matches the GID, then are conferred; otherwise, the process has to the object. Linux performs access control by assigning objects a that specifies which access modes-read, write, or execute-are to be granted to processes with owner, group, or world access. Thus, the owner of an object might have full read, write, and execute access to a file; other users in a certain group might be given read access but denied write access; and everybody else might be given no access at all.

The only exception is the privileged UID. A process with this special UID is granted automatic access to any object in the system, bypassing normal access checks. Such processes are also granted permission to perform privileged operations, such as reading any physical memory or opening reserved network sockets. This mechanism allows the kernel to prevent normal users from accessing these resources: most of the kernel's key internal resources are implicitly owned by the root UID. Linux implements the standard UNIX setuid mechanism described in Section A.3.2. This mechanism allows a program to run with privileges different from those of the user running the program. For example, the lpr program (which submits a job to a print queue) has access to the system's print queues even if the user running that program does not. The UNIX implementation of setuid distinguishes between a process's real and effective UID. The real UID is that of the user running the program; the effective UID is that of the file's owner.

History Of Windows Through Windows 10:

In the mid-1980s, Microsoft and IBM cooperated to develop the OS/2 operating system, which was written in assembly language for single-processor Intel 80286 systems. In 1988, Microsoft decided to make a fresh start and to develop a "new technology" (or NT) portable operating system that supported both the OS/2 and POSIX application-programming interfaces (APis). In October 1988, Dave Cutler, the architect of the DEC VAX/VMS operating system, was hired and given the charter of building this new operating system. Originally, the team planned to use the OS/2 API as NTs native environment, but during development, NT was changed to use the 32-bit Windows API (or Win32 API), reflecting the popularity of Windows 3.0. The first versions of NT were Windows NT 3.1 and Windows NT 3.1 Advanced Server. (At that time, 16-bit Windows was at version 3.1.) Windows NT Version 4.0 adopted the Windows 95 user interface and incorporated Internet Web-server and Web- browser software. In addition, user-interface routines and all graphics code were moved into the kernel to improve performance, with the side effect of decreased system reliability. Although previous versions of NT had been ported to other microprocessor architectures, the Windows 2000 version, released in February 2000, supported only Intel (and compatible) processors due to marketplace factors.

Windows 2000 incorporated significant changes. It added Active Directory (an X.SOO-based directory service), better networking and laptop support, support for plug-and-play devices, a distributed file system, and support for more processors and more memory. In October 2001, Windows XP was released as both an update to the Windows 2000 desktop operating system and a replacement for Windows 95/98. In 2002, the server versions of Windows XP became available (called Windows .Net Server). Windows XP updated the graphical user interface (GUI) with a visual design that took advantage of more recent hardware advances and many new Numerous features were added to automatically repair problems in applications and the operating system itself. As a result of these changes, Windows XP provides better networking and device experience (including zero-configuration wireless, instant messaging, streaming media, and digital photography /video), dramatic performance improvements for both the desktop and large multiprocessors, and better reliability and security than earlier Windows operating systems. Windows XP uses a client-server architecture (like Mach) to implement multiple operating-system personalities, such as Win32 API and POSIX, with user-level processes called subsystems. The subsystem architecture allows enhancements to be made to one operating-system personality without affecting the application compatibility of any others.

Windows XP is a multiuser operating system, supporting simultaneous access through distributed services or through multiple instances of the graphical user interface via the Windows terminal server. The server versions of Windows XP support simultaneous

terminal server sessions from Windows desktop systems. The desktop versions of terminal server multiplex the keyboard, mouse, and monitor between virtual terminal sessions for each logged-on user. This feature, called fast user switching, allows users to pre-empt each other at the console of a PC without having to log off and onto the system.

Windows XP was the first version of Windows to ship a 64-bit version. The native NT file system~ and many of the Win32 APIs have always used 64-bit integers where appropriate-so the major extension to 64-bit in Windows XP was support for large addresses.

There are two desktop versions of Windows XP. Windows XP Professional is the premium desktop system for power users at work and at home. Windows XP Personal provides the reliability and ease of use of Windows XP but lacks the more advanced features needed to work seamlessly with Active Directory or run POSIX applications.

The mernbers of the Windows .Net Server family use the same core components as the desktop versions but add a range of features needed for uses such as webserver farms, print/ file servers, clustered systems, and large datacenter machines. The large datacenter machines can have up to 64GB of memory and 32 processors on IA32 systems and 128 GB and 64 processors on IA64 systems.

Microsoffs design goals for Windows XP included security, reliability, Windows and POSIX application compatibility, high performance, extensibility, portability, and international support. We discuss each of these goals in the following sections.

Security:

Windows XP goals required more than just adherence to the design standards that had enabled Windows NT 4.0 to receive a C-2 security classification from the U.S. government (which signifies a moderate level of protection from defective software and malicious attacks). Extensive code review and testing were combined with sophisticated automatic analysis tools to identify and investigate potential defects that might represent security vuh1erabilities.

Reliability

Windows 2000 was the most reliable, stable operating system Microsoft had ever shipped to that point. Much of this reliability came from maturity in the source code, extensive stress testing of the and automatic detection of many serious errors in drivers. The requirements for Windows XP were even more stringent. Microsoft used extensive manual and automatic code review to identify over 63,000 lines in the source files that might contain issues not detected by testing and then set about reviewing each area to verify that the code was indeed correct.

Windows XP extended driver verification to catch more subtle bugs, the facilities for catching programming errors in user-level code, and subjected third-party applications, drivers, and devices to a rigorous certification process. Further , Windows XP added

facilities for monitoring the health of the PC, including downloading fixes for problems before they are encountered by users. The perceived reliability of Windows XP was also improved by making the graphical user interface easier to use through better visual design, simpler menus, and measured improvements in the ease with which users can discover how to perform common tasks..