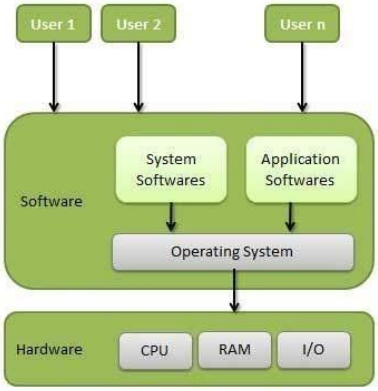
UNIT-1

What is operating system?

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

An **Operating System (OS)** is software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. An operating system is 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.

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.



Operating System

Following are some of important functions of an operating System.

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

The Operating System is a program with the following features;

- An operating system is a program that acts as an interface between the software and the computer hardware.
- It is an integrated set of specialized programs used to manage overall resources and operations of the computer.

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

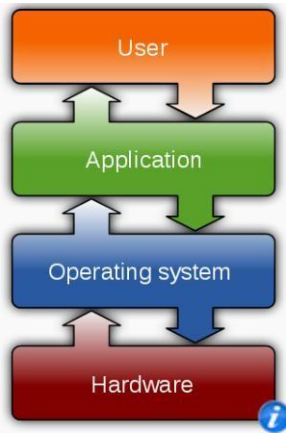


Figure 1: Operating System

Objectives of Operating System

The objectives of the operating system are −

- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

Characteristics of Operating System

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

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

Processor Management − Allocates the processor (CPU) to a process and deallocates the processor when it is no longer required.

Device Management − Keeps track of all the devices. This is also called I/O controller that decides which process gets the device, when, and for how much time.

File Management − Allocates and de-allocates the resources and decides who gets the resources

Security − Prevents unauthorized access to programs and data by means of passwords and other similar techniques.

Job Accounting − Keeps track of time and resources used by various jobs and/or users.

Control Over System Performance − Records delays between the request for a service and from the system.

Interaction with the Operators − Interaction may take place via the console of the computer in the form of instructions. The Operating System acknowledges the same, does the corresponding action, and informs the operation by a display screen.

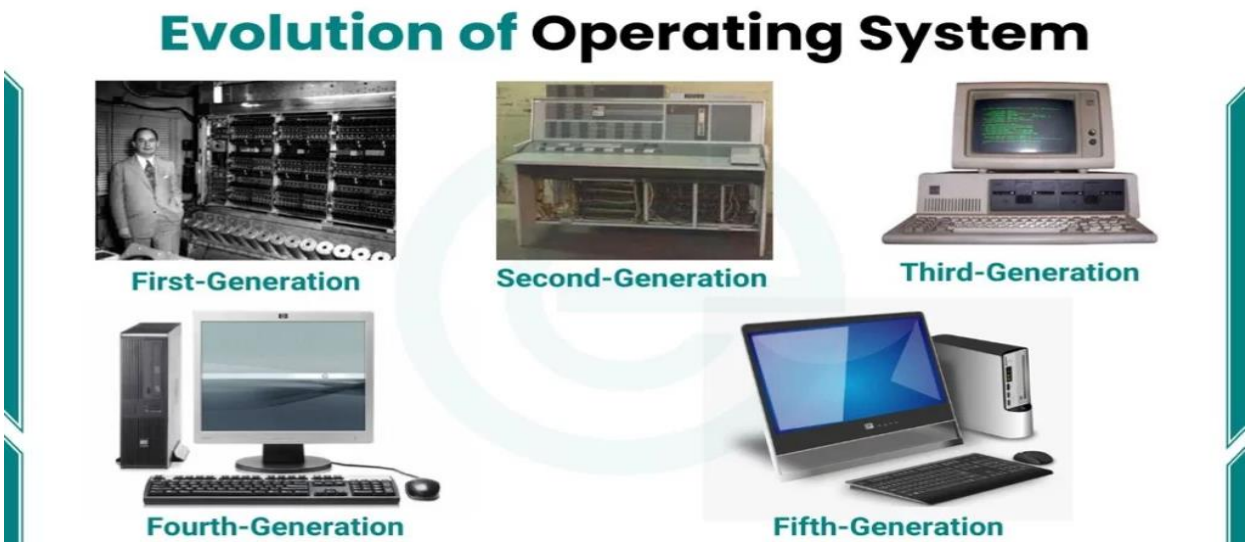
Error-detecting Aids − Production of dumps, traces, error messages, and other debugging and error-detecting methods.

Coordination Between Other Software and Users − Coordination and assignment of compilers, interpreters, assemblers, and other software to the various users of the computer systems.

History of Operating System

An operating system is a type of software that acts as an interface between the user and the hardware. It is responsible for handling various critical functions of the computer and utilizing resources very efficiently so the operating system is also known as a resource manager. The operating system also acts like a government because just as the government has authority over everything, similarly the operating system has authority over all resources. Various tasks that are handled by OS are file management, task management, garbage management, memory management, process management, disk management, I/O management, peripherals management, etc.

Generations of Operating Systems



First Generation (1945s-1955s)

Technology: Vacuum tubes.
OS type: No operating system initially; early systems were managed by operators. Batch processing systems were later developed to automate job handling.
Key feature: Computers were large, expensive, and programs were run one at a time.

Second Generation (1955-1965)

Technology: Transistors replaced vacuum tubes.
OS type: Batch processing systems, like GMOS, became more common to improve efficiency by grouping similar jobs.
Key feature: Transistors made computers smaller, cheaper, and more reliable, leading to the creation of mainframes.

Third Generation (1965-1980)

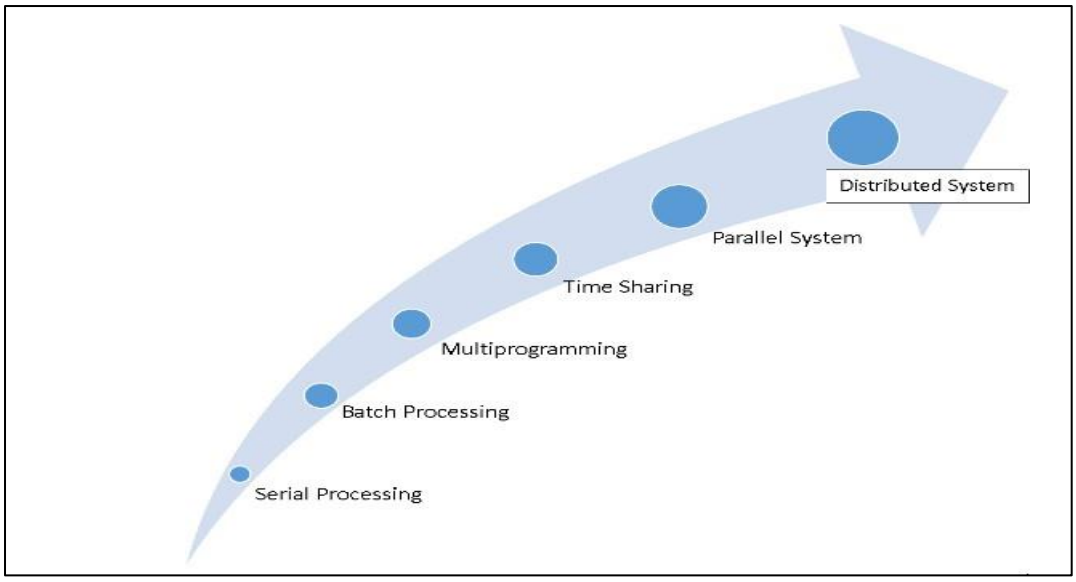
Technology: Integrated circuits (ICs) were introduced, allowing for smaller, faster, and more powerful computers.
OS type: Time-sharing operating systems, such as UNIX, enabled multiple users to interact with a single computer simultaneously. Multiprogramming also became standard.
Key feature: Increased user accessibility and the ability for simultaneous interaction with the computer.

Fourth Generation (1980s-Present)

Technology: Microprocessors and large-scale integrated circuits enabled the development of personal computers.
OS type: Operating systems for personal computers became prevalent, with the rise of GUIs like Microsoft Windows and macOS. Networking capabilities also became a standard feature.
Key feature: The shift to user-friendly personal computers, graphical interfaces, and the internet.

Evolution of of operating System / Types of Operating System

Evolution of OS since 1950 described in detail in this article. Here we will discuss six main operating system types evaluated over the past 70 years. Following diagram shows the evolution of the operating system.



Evolution of Operating System.

1) Serial Processing

History of the operating system started in 1950. Before 1950, the programmers directly interact with the hardware there was no operating system at that time. If a programmer wishes to execute a program on those days, the following serial steps are necessary Type the program or punched card.

- Convert the punched card to a card reader.
- Submit to the computing machine, is there any errors, the error was indicated by the lights.
- The programmer examined the register and main memory to identify the cause of an error
- Take outputs on the printers.
- Then the programmer ready for the next program.

Drawback:

This type of processing is difficult for users, it takes much time and the next program should wait for the completion of the previous one. The programs are submitted to the machine one after one, therefore the method is said to be **serial processing**.

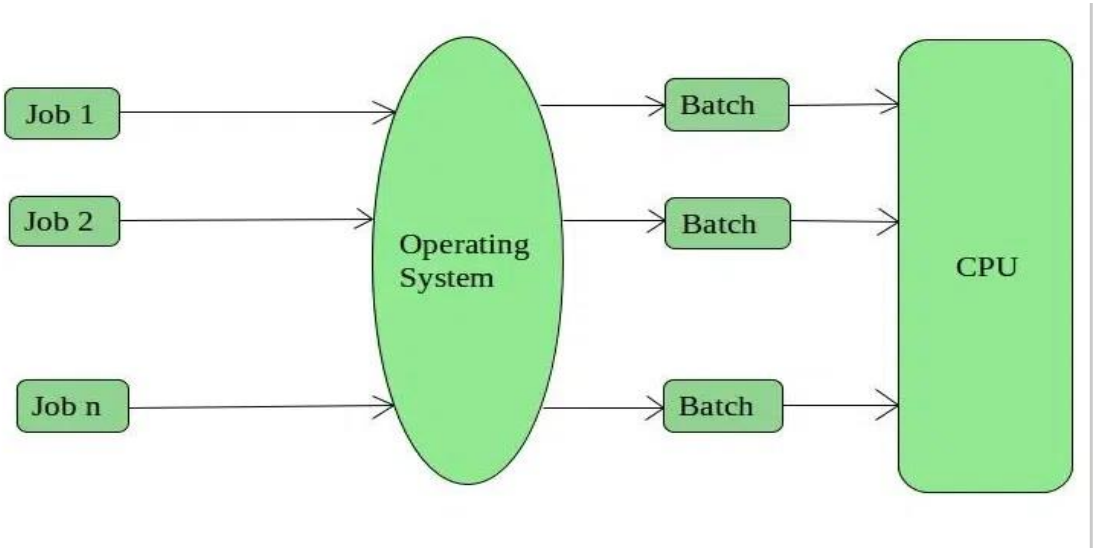
2) Batch Processing

Before 1960, it is difficult to execute a program using a computer because of the computer located in three different rooms, one room for the card reader, one room for executing the program and another room for printing the result.

The user/machine operator runs between three rooms to complete a job. We can solve this problem by using **batch processing**.

In batch processing technique, the same type of jobs batch together and execute at a time. The carrier carries the group of jobs at a time from one room to another.

Therefore, the programmer need not run between these three rooms several times.



This type of operating system does not interact with the computer directly.
There is an operator which takes similar jobs having the same requirements and groups them into batches.
It is the responsibility of the operator to sort jobs with similar needs.
Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.

Advantages of Batch Operating System

1. Multiple users can share the batch systems.
2. The idle time for the batch system is very less.
3. It is easy to manage large work repeatedly in batch systems.

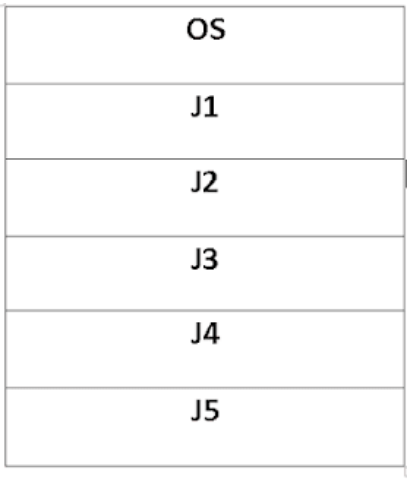
Disadvantages of Batch Operating System

1. Batch systems are hard to debug.
2. It is sometimes costly.
3. The other jobs will have to wait for an unknown time if any job fails.
4. In batch operating system the processing time for jobs is commonly difficult to accurately predict while they are in the queue.

Examples of Batch Operating Systems: Payroll Systems, Bank Statements, etc.

3) Multiprogramming

Multiprogramming is a technique to execute the number of programs simultaneously by a single processor. In multiprogramming, a number of processes reside in main memory at a time. The OS (Operating System) picks and begins to execute one of the jobs in main memory. Consider the following figure, it depicts the layout of the multiprogramming system. The main memory consisting of 5 jobs at a time, the CPU executes one by one.



Jobs in OS

In the non-multiprogramming system, the CPU can execute only one program at a time, if the running program is waiting for any I/O device, the CPU becomes idle so it will effect on the performance of the CPU.

But in a multiprogramming environment, if any I/O wait happened in a process, then the CPU switches from that job to another job in the job pool. So, the CPU is not idle at any time.

Advantages of Multi-Programming Operating System

1. Can get efficient memory utilization.
2. CPU is never idle so the performance of CPU will increase.
3. The throughput of CPU may also increase.
4. In the non-multiprogramming environment, the user/program has to wait for CPU much time. But waiting time is limited in multiprogramming.
5. It helps in reducing the response time.

Disadvantages of Multi-Programming Operating System

There is not any facility for user interaction of system resources with the system.

4) Time Sharing System

Time-sharing or **multitasking** is a logical extension of multiprogramming. Multiple jobs are executed by the CPU switching between them. The CPU scheduler selects a job from the ready queue and switches the CPU to that job. When the time slot expires, the CPU switches from this job to another.

In this method, the CPU time is shared by different processes. So, it is said to be "**Time-Sharing System**". Generally, time slots are defined by the operating system.

Advantages of Time-Sharing OS:

1. The main **advantage of the time-sharing system** is efficient CPU utilization. It was developed to provide interactive use of a computer system at a reasonable cost. A time shared OS uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.
2. Another advantage of the time-sharing system over the batch processing system is, the user can interact with the job when it is executing, but it is not possible in batch systems.
3. Each task gets an equal opportunity.

4. Fewer chances of duplication of software.
5. CPU idle time can be reduced.

Disadvantages of Time-Sharing OS:

1. Reliability problem
2. Data communication problem

5) Parallel System

There is a trend multiprocessor system, such system have more than one processor in close communication, sharing the computer bus, the clock, and sometimes memory and peripheral devices.

These systems are referred to as "Tightly Coupled" system. Then the system is called **a parallel system**. In the parallel system, a number of processors are executing there job in parallel.

Advantages of Parallel Operating System

1. It increases the throughput.
2. By increasing the number of processors (CPU), to get more work done in a shorter period of time.
3. It saves time and allows the execution of applications simultaneously
4. Solves the large complex problem of operating system
5. Multiple resources can be used simultaneously
6. Has a larger memory for the allocation of resources and tasks
7. Faster as compared to another operating system

Disadvantages of Parallel Operating System

1. The architecture of the parallel operating system is complex
2. High cost since more resources are used for synchronization, data transfer, thread, and communication. In the case of clusters, better cooling techniques are required.
3. Huge power consumption.
4. High maintenance.

6) Distributed System

In a **distributed operating system**, the processors cannot share a memory or a clock, each processor has its own local memory. The processor communicates with one another through various communication lines, such as high-speed buses. These systems are referred to as "Loosely Coupled" systems.

Advantages of Distributed Operating System:

1. If a number of sites connected by high-speed communication lines, it is possible to share the resources from one site to another site, for example, s1 and s2 are two sites. These are connected by some communication lines. The site s1 having a printer, but

   the site does not have any print. Then the system can be altered without moving from s2 to s1. Therefore, resource sharing is possible in the distributed operating system.
2. A big computer that is partitioned into a number of partitions, these sub-partitions are run concurrently in distributed systems.
3. If a resource or a system fails in one site due to technical problems, we can use other systems/resources in some other sites. So, the reliability will increase in the distributed system.
4. Failure of one will not affect the other network communication, as all systems are

independent from each other
5. These systems are easily scalable as many systems can be easily added to the network

Disadvantages of Distributed Operating System:

1. Failure of the main network will stop the entire communication
2. These types of systems are not readily available as they are very expensive

7) Mobile Operating System

A mobile operating system (OS) is software that enables smartphones, tablets and other devices to run applications and programs. It provides an interface between the device's hardware components and software functions.

An operating system (OS) for mobile devices manages the basic functions. It runs apps, controls memory, and connects to networks. These systems provide an easy-to-use interface.

They're designed for smartphones, tablets, and wearable tech. Mobile OSes let you multitask, browse the web and download apps. Popular options include iOS from Apple, Android from Google, and Huawei's HarmonyOS. Mobile operating systems power modern gadgets. With them, phones become portable computers, communication tools, and entertainment hubs.

Key Features of a Mobile Operating System

1. **User Interface (UI):** Touch inputs of Graphical User Interface (GUI) provided by mobile OS are optimized. This is where users can use touch gestures, in other words, swiping, tapping, and pinching, to interact with their gadgets.

2. **Multitasking:** It helps in running of many apps at the same time but what is more we can quickly switch between them without any hindrance. Such offloading is for applications that are not currently used actively.

3. **Connectivity:** It provides a variety of connections such as cellular, Wi-Fi, Bluetooth, NFC (Near Field Communication) and others to facilitate the communication of the device with other devices and networks.

4. **Application Management:** Is a platform that has its own app marketplace or store which the users utilize to browse, install, run and updates the applications exclusively for that platform.

5. **Resource Management:** Efficiently allocates hardware resources like the CPU ,ram , and battery by achieving a balance between performance and battery life.

Types of Mobile Operating System

The mobile world has seen many operating systems. Some are big and powerful. Others are small and niche. Here are the main types of mobile OS :

1. **Android:** Made by Google, Android is the most used mobile OS worldwide. It's an open-source system built on Linux code. Android is made mostly for touchscreen phones and tablets. Being open allows companies to customize Android for their gadgets. That leads to many different Android devices.

2. **Apple Inc. created iOS**, the operating system for iPhone, iPad, and iPod Touch. Its smooth interface and tight Apple ecosystem integration are hallmarks. However, Apple's total

control over hardware and software limits customization options, unlike Android's open platform.

3. Huawei developed HarmonyOS (Hongmeng OS in China): work across diverse devices like smartphones, wearables, laptops, smart home gadgets. By offering a unified ecosystem, Huawei aims to navigate US government restrictions on its business.
4. KaiOS is a basic mobile OS: It powers basic phones without touchscreens. KaiOS is based on discontinued Firefox OS. KaiOS supports 4G, GPS, and apps like Facebook and WhatsApp. It offers strong capabilities for non-smartphone mobile devices.

6. **Tizen** is a Linux-based mobile OS developed by Samsung. It was created with Intel and the Tizen Association. Tizen powers various Samsung products.

Need/Importance of an Operating System

Operating System is needed to enable the user to design the application or software without concerning the details of the computer's internal structure. In general the boundary between the hardware & software is transparent to the user.

OS as a platform for Application programs: Operating system provides a platform, on top of which, other programs, called application programs can run. These application programs help the users to perform a specific task easily. It acts as an interface between the computer and the user. It is designed in such a manner that it operates, controls and executes various applications on the computer.

Managing Input-Output unit: Operating System also allows the computer to manage its own resources such as memory, monitor, keyboard, printer etc. Management of these resources is required for an effective utilization. The operating system controls the various system input-output resources and allocates them to the users or programs as per their requirement.

Consistent user interface: Operating System provides the user an easy-to-work user interface, so the user doesn't have to learn a different UI every time and can focus on the content and be productive as quickly as possible. Operating System provides templates, UI components to make the working of a computer, really easy for the user.

Multitasking: Operating System manages memory and allow multiple programs to run in their own space and even communicate with each other through shared memory. Multitasking gives users a good experience as they can perform several tasks on a computer at a time.

Process Management: - The process management module of an Operating System takes care of the creation & deletion of processes, scheduling of various system resources to the different process requesting them, & providing mechanism for synchronization & communication among processes.

Memory Management: - The memory management module of an Operating System takes care of the allocation & reallocation of memory space to the various program in need of this resource.

File Management: - computer use a lot of data & programs, which are, stored on secondary storage devices. File management functions of an Operating System. Involves

keeping track of all different files & maintaining the integrity of data stored in the files including file directory structure.

Security: - The security modules of an Operating System protect the resources & information of a computer system against destruction& unauthorized access.

Command Interpretation: -The Command Interpretation module of an Operating System takes care of interpreting of user commands, & directing the system resources to handle the requests. With this mode of interaction with the system, the user is usually not too concerned with the hardware details of the system.

Device Management: - coordination & control of various input & output devices is an important function of the Operating System. This involves receiving the request for I/O interrupts, & communicating back to the requesting process.

Job Control: - When the user wants to run an application program, he must communicate with the Operating System telling it what to do. He does this using Operating System job control language or JCL. JCL consists of a number of Operating Systems commands, called system commands that control the functioning of the Operating System.

Operating system as a resource Manager: Operating system is the central controller of the computer to manage, accept and grant resource request of various processes.

Single User & Multi User Operating System

The main difference between single user and multiuser operating system is that in a single user operating system, only one user can access the computer system at a time while in a multiuser operating system, multiple users can access the computer system at a time.

A single user operating system provides the facilities to be used on one computer by only one user. In other words, it supports one user at a time. However, it may support more than one profiles. Single keyboard and single monitor are used for the purpose of interaction. The most common example of a single user operating system is a system that is found in a typical home computer. Also, see the difference between Android and Windows.

On the other hand, a multi-user operating system has been designed for more than one user to access the computer at one time. Generally, a network is laid down, so that a computer can be remotely used. Mainframes and minicomputers work on multi-user operating systems. These operating systems are complex in comparison to single user operating systems. Each user is provided with a terminal and all these terminals are connected to the main computer. In a multi-user environment, it is very important to balance the requirements of the users, as the resources of the main computer are shared among the users. For example, unix, linux, windows server os etc.

Difference between Single User and Multi-User Operating System:

Point of Difference	Single User	Multi-User

Definition	A single user operating system provides facilities to be used on one computer by only one user.	A multi-user operating system has been designed for more than one user to access the computer at the same or different time.
Types	Single user, single task: A single task is performed by one user at a time. Example- The Palm OS for Palm handheld computers. Single user, multi-task: Several programs are run at the same time by a single user. For example- Microsoft Windows.	Time sharing systems: These systems are multi-user systems in which CPU time is divided among the users. The division is made on the basis of a schedule. E.g. Unix or Linux Most batch processing systems for the mainframe computers can also be considered as 'multi user.'
Attributes	Simple	Complex
Performance	Only one task at one time gets performed.	Schedules different tasks for performance at any rate
Super User	A super user gets all the powers of maintaining the system and making changes to ensure the system runs smoothly.	Super user does not exist when it comes to a multi-user operating system as each entity has control over their working.
Examples	Windows, Apple, Mac etc	Unix, Linux and mainframes such as the IBM AS400.

ELEMENTS/COMPONENTS/PARTS/LAYERS OF OPERATING SYSTEM

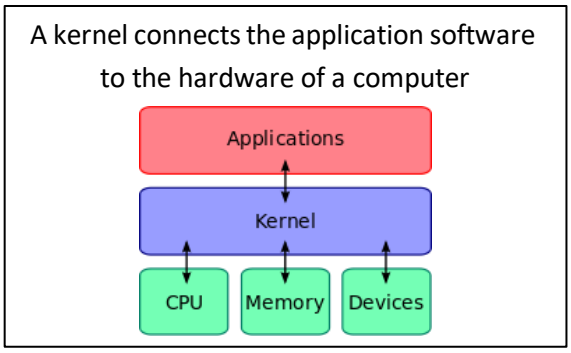
This topic gives an overview of the various elements/layers which make up an operating system whether it can be Windows, Linux or Mac, serves the purpose of giving us, the human user, a means to interact with the computer in a meaningful way.

The Kernel.

A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware

(CPU, disk memory etc). The main tasks of the kernel are:

- Process management
- Device management
- Memory management
- Interrupt handling
- I/O communication
- File system...etc.



The operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.

The kernel is the heart of the operating system. Amongst its responsibilities are ensuring that each running process is given a fair amount of time to execute while a controlling the amount of resources each process can use.

During ‘‘normal’’ operations of a computer system, some portions of the operating system remain in main memory to provide services for critical
Operations, such as dispatching, interrupt handling, or managing (critical) resources. These portions of the OS are collectively called the *kernel*.

Kernel (*remain)* = OS – transient components (*comes and goes)*

Memory Management.

The name of this layer gives you a good idea what it is all about. It is the responsibility of this layer to share your computers physical memory among the processes which want to use it. It also has to manage such situations where there may not be enough physical memory to share out.

Memory must be shared between the OS and an application program. The OS must manage the allocation of memory to processes and control the memory management hardware that determines which memory locations a process may access.

Input/Output.

On this layer all the physical communication between your computer’s hardware, such as disk drives, keyboards, mouses, screens and so on, takes place.

File Management.

Again the name of this layer may give you a clue as to what it does. It is the job of this layer to control how the files on your computers hard drive are stored and accessed by any application seeking to use them.

Computers process information that must be transmitted, processed, or stored. File systems are an abstract organized collection of file system objects. The OS provides

primitives to manipulate these objects.

The User Interface.

The other element, or layer as we have been calling them, of an operating system is the User Interface. This layer is probably the easiest of all to understand since it is the first thing you see when your operating system has logged you in. It is the job of this layer to provide a means for the user to actually interact with the rest of the layers and as such the system as a whole.

Keep in mind there are two different types of User interfaces. The first one is probably the one you are most familiar with, the graphical user interface, which is where you see windows and icons for each of your files and so on.

The second is a command line interface, or text based interface where a user would interact with the system using text based commands.

Process Management

A *process* is an executing program. It has its code, data, a certain set of resources allowed to it, and one or more flows of execution through the code. The OS manages the allocation of resources to these processes, and also provides system calls to manage these processes.

Device Management

Information is sent through a computer's input and output devices. Processes access these devices using the *system call* interface. The OS tries to manage said devices in a manner that makes them efficiently shared among all processes requiring them. A *system call* is a programming interface to the services provided by the OS, typically written in C/C++. (System call is an interface between process and operating system.)

Interrupt

In the operating system, interrupts are essential because they give a reliable technique for the OS to communicate & react to their surroundings. An interrupt is nothing but one kind of signal between a device as well as a computer system otherwise from a program in the computer that requires the OS to leave and decide accurately what to do subsequently. Whenever an interrupt signal is received, then the hardware of the computer puts on hold automatically whatever computer program is running presently, keeps its status & runs a computer program which is connected previously with the interrupt.

Multitasking

It describes the working of several independent computer programs on a similar computer system. Multitasking in an OS allows an operator to execute one or more computer tasks at a time. Since many computers can perform one or two tasks at a time, usually this can be done with the help of time-sharing, where each program uses the time of a computer to execute.

Networking

Networking can be defined as when the processor interacts with each other through communication lines. The design of communication-network must consider routing, connection methods, safety, the problems of opinion & security.

Presently most of the operating systems maintain different networking techniques, hardware, & applications. This involves that computers that run on different operating systems could be included in a general network to share resources like data, computing, scanners, printers, which uses the connections of either wired otherwise wireless.

Security

If a computer has numerous individuals to allow the immediate process of various processes, then the many processes have to be protected from other activities. This system security mainly depends upon a variety of technologies that work effectively. Current operating systems give an entrée to a number of resources, which are obtainable to work the software on the system, and to external devices like networks by means of the kernel. The operating system should be capable of distinguishing between demands which have to be allowed for progressing & others that don't need to be processed. Additionally, to permit or prohibit a security version, a computer system with a high level of protection also provides auditing options. So this will allow monitoring the requests from accessibility to resources

The Operating System as Resource Manager

A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions. The OS is responsible for managing these resources. Can we say that it is the OS that controls the movement, storage, and processing of data? From one point of view, the answer is yes: By managing the computer's resources, the OS is in control of the computer's basic functions. But this control is exercised in a curious way. Normally, we think of a control mechanism as something external to that which is controlled, or at least as something that is a distinct and separate part of that which is controlled. (For example, a residential heating system is controlled by a thermostat, which is separate from the heat generation and heat distribution apparatus.) This is not the case with the OS, which as a control mechanism is unusual in two respects:

• The OS functions in the same way as ordinary computer software; that is, it is a program or suite of programs executed by the processor.

• The OS frequently relinquishes control and must depend on the processor to allow it to regain control.

Like other computer programs, the OS provides instructions for the processor. The key difference is in the intent of the program. The OS directs the processor in the use of the other system resources and in the timing of its execution of other programs. But in order for the processor to do any of these things, it must cease executing the OS program and execute other programs. Thus, the OS relinquishes control for the processor to do some "useful" work and then resumes control long enough to prepare the processor to do the next piece of work. The mechanisms involved in all this should become clear as the chapter proceeds.

Figure 2.2 suggests the main resources that are managed by the OS.A portion of the OS is in main memory. This includes the **kernel**, or **nucleus**, which contains the most frequently used functions in the OS and, at a given time, other portions of the OS currently in use. The remainder of main memory contains user programs and data. The allocation of this resource (main memory) is controlled jointly by the OS and memory management hardware in the processor, as we shall see. The OS decides when an I/O

device can be used by a program in execution and controls access to and use of files. The processor itself is a resource, and the OS must determine how much processor time is to be devoted to the execution of a particular user program. In the case of a multiple-processor system, this decision must span all of the processors.

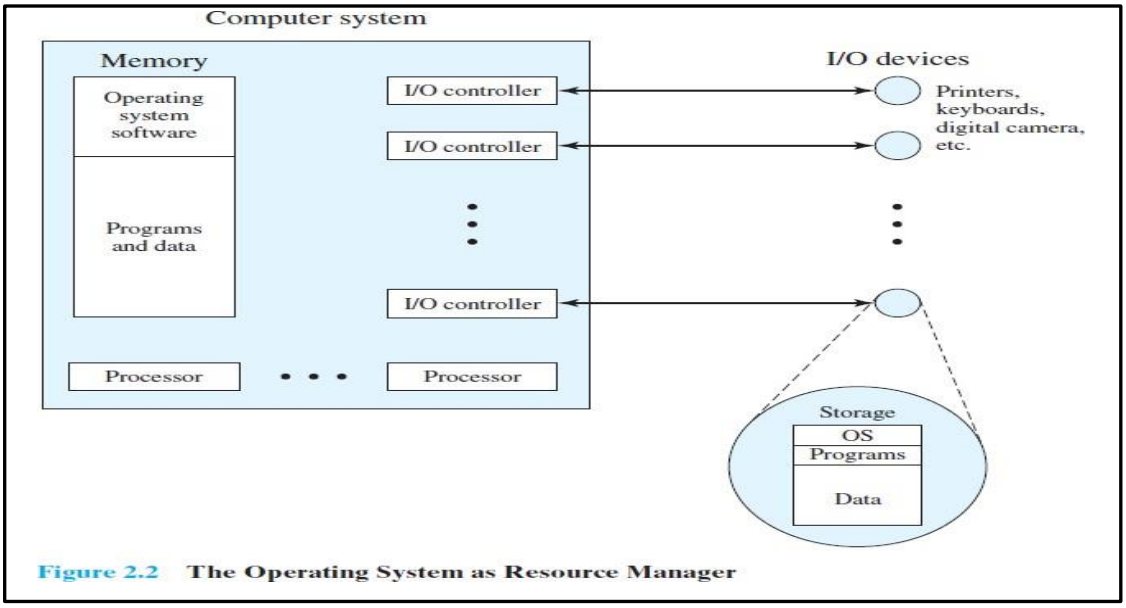


Figure 2.2 The Operating System as Resource Manager

The Operating System is a manager of system resources. A computer system has many resources as stated above. Since there can be many conflicting requests for the resources, the Operating System must decide which requests are to be allocated resources to operate the computer system fairly and efficiently. Here we present a framework of the study of Operating System based on the view that the Operating System is manager of resources.

The primary view is that the Operating System is a collection of programs designed to manage the system's resources, namely, memory, processors, peripheral devices, and information. It is the function of Operating System to see that they are used efficiently and to resolve conflicts arising from competition among the various users. The Operating System must keep track of status of each resource; decide which process is to get the resource, allocate it, and eventually reclaim it.

The major functions of each category of Operating System are;

Memory Management Functions

To execute a program, it must be mapped to absolute addresses and loaded into memory. As the program executes, it accesses instructions and data from memory by generating these absolute addresses. In multiprogramming environment, multiple programs are maintained in

the memory simultaneously. The Operating System is responsible for the following memory management functions:

 Keep track of which segment of memory is in use and by whom.

 Deciding which processes are to be loaded into memory when space becomes available. In multiprogramming environment it decides which process gets the available memory, when it gets it, where does it get it, and how much.

 Allocation or de-allocation the contents of memory when the process request for it otherwise reclaim the memory when the process does not require it or has been terminated.

Processor/Process Management Functions

A process is an instance of a program in execution. While a program is just a passive entity, process is an active entity performing the intended functions of its related program. To accomplish its task, a process needs certain resources like CPU, memory, files and I/O devices. In multiprogramming environment, there will a number of simultaneous processes existing in the system. The Operating System is responsible for the following processor/ process management functions:

 Provides mechanisms for process synchronization for sharing of resources amongst concurrent processes.

 Keeps track of processor and status of processes. The program that does this has been called the traffic controller.

 Decide which process will have a chance to use the processor; the job scheduler chooses from all the submitted jobs and decides which one will be allowed into the system. If multiprogramming, decide which process gets the processor, when, for how much of time. The module that does this is called a process scheduler.

 Allocate the processor to a process by setting up the necessary hardware registers. This module is widely known as the dispatcher.

 Providing mechanisms for deadlock handling.

 Reclaim processor when process ceases to use a processor, or exceeds the allowed amount of usage.

I/O Device Management Functions

An Operating System will have device drivers to facilitate I/O functions involving I/O devices. These device drivers are software routines that control respective I/O devices through their controllers. The Operating System is responsible for the following I/O Device Management Functions:

 Keep track of the I/O devices, I/O channels, etc. This module is typically called I/O traffic controller.

 Decide what is an efficient way to allocate the I/O resource. If it is to be shared, then decide who gets it, how much of it is to be allocated, and for how long. This is called I/O scheduling.

 Allocate the I/O device and initiate the I/O operation.

 Reclaim device as and when its use is through. In most cases I/O terminates automatically.

Information Management Functions

 Keeps track of the information, its location, its usage, status, etc. The module called a file system provides these facilities.

 Decides who gets hold of information, enforce protection mechanism, and provides for information access mechanism, etc.

 Allocate the information to a requesting process, e.g., open a file.

 De-allocate the resource, e.g., close a file.

Network Management Functions

An Operating System is responsible for the computer system networking via a distributed environment. A distributed system is a collection of processors, which do not share memory, clock pulse or any peripheral devices. Instead, each processor is having its own clock pulse, and RAM and they communicate through network. Access to shared resource permits increased speed, increased functionality and enhanced reliability. Various networking protocols are TCP/IP (Transmission Control Protocol/ Internet Protocol), UDP (User Datagram Protocol), FTP (File Transfer Protocol), HTTP (Hyper Text Transfer protocol), NFS (Network File System) etc.

UNIT-2: FILE MANAGEMENT

1) Overview of File

Computers can store information on different storage media, such as magnetic disk, taps, pen drive, hard disk drive (HDD) and compact disk. The physical storage is converted into the logical storage unit by operating system. The logical storage unit is said to be a file. In general, File is a collection of data. File is a mapped by the operating system on to physical device. A file is a collection of similar 'record' with a common name. A record is collection of related fields that can be treated as a unit by some applications programs. A field is the some basic element of data. Any individual field contains single value. A database is a collection of related data.

For example

Table-1: Records in a table

Student no.	Sub 1	Sub 2	Sub3	Sub 4	Fail/pass
1001	36	45	85	96	Pass
1002	23	69	96	78	Fail
1003	99	86	47	52	Pass
1004	22	36	14	99	Fail

Here, student number, subject marks, pass/fail are fields or data elements. The collection of these data element is called record.

1004	22	36	14	99	Fail

Collection of these records called data file. A database may contain all of the information related to an organization or project. So, a database consisting of different types of data files.

2) OPERATIONS ON DATABASE FILES

Retrieve –all: this command retrieves all the records of a file. It must be useful to perform the operation on the entire file at a time. For example we want to view all the records in student database file.

Retrieve-one: this command retrieves one record of a file at a time. Interactive, transitive oriented applications need this command. Suppose we want to modify or delete a single record at a time then this command is used.

Retrieve-next: this command retrieves the next record which is in the next record to the recently received one. Some interactive applications like filling in forms may require this command the most.

Retrieve-previous: this command retrieves the previous record which is in the previous record to the recently retrieved one.

Insert-one: insert a new record to the existing database file.

Delete-one: delete an existing record form an existing file specified by the user.

Update-one: Retrieve a record, update one or more of it's fields, and rewrites the updated record back into the file.

Retrieve-few: Retrieve a number of records. For example an application or user may write to retrieve all records that specify a certain set of criteria.

3) OPRATIONS ON PROGRAMMABLE/OTHER FILES:

The basic operations of the programmable files are creating a file, writing a file, reading a file, repositioning within a file, deleting a file, and truncating a file. The operating system provides system calls to create, write, read, repositioning, delete, and truncate files.

Creating a file: Two steps are needed to create a file. First one is to check whether the space is available or not. If the space is available then made an entry for a new file must be made in the directory. The entry includes name, size, and path of the file.

Writing a file: to write a file, we have to know two things. One is name of the file and second is the information or the data to be written in the file. The system searches the entire given location for the file. If the file is found, the system must keep a write pointer to the location in the file where the next write is to take place.

Reading a file: To read a file, first of all we search the directories for the file. If the file is found, the system needs to keep a read pointer to the location in the file where the new read is to take place. Once the read has taken place, the read pointer is updated.

Repositioning within a file: This file operation is also known as file seek. The directory is searched for the appropriate entry and current file position is set to a given value.

Deleting a file: To delete a file, first of all we search directories for the file name, then release the file space and erase directory entry.

Truncating a file: To truncate a file, remove the file contents only. But the attributes are as it is.

4) ATTRIBUTES/PROPERTIES OF FILE

There are various file attributes available.

Name, type, location, size, protection, time, date, user identification etc

Name: a file is named. For the convenience of the user. And is referred to by its name. A name is usually a string of characters. The symbolic file name is the only information kept in human readable form.

Type: files are of so many types. The type is depending on the extension of the file.

For example:

- **.exe: executable file.**
- **.obj: object file.**
- **.src: source file.**

Generally, the name of the file split into two parts one is name and second is extension. The file type is depending on the extension of the file. For example in MS-DOS a name can allowed up to 8 character followed by a period and terminated by an up to three characters of extension.

FILE TYPE	**EXTENSION**	**PURPOSE/FUNCTION**
Executable	.exe .com .bin Or none	Ready to run(or) Ready to run machine Language program.
Source code	.c .cpp .pas .asm .f77	Source code in various languages
Batch	.bat .sh	Commands to the command interpreter.
Text	.txt .doc	Textual data Documents
Word processors	.wp .rrf .etc	Various word processors.
Library	.lib .a	Libraries of routines for programmers
Point or view	.ps .dvi .sif	ASCII or binary file in a format for printing viewing.
Archive	.arc .zip .tor	Grouped files Compressed files or Archiving or storage.

Location: this information is a pointer to a device and to the location of the file on what device.

Size: the current size of the file. (In bytes, blocks)

Protection: it specifies the access-control to the information. It controls who can read, write, execute and so on.

Time: it specifies time to creation/modification.

Date: it specifies file created/modified date.

User identification: this is useful for protection security and usages monitoring.

5) FILE MANAGEMENT SYSTEM

A file management system (FMS) is a collection of system software programs that provides service to the user. For example that user want to create a file, delete file, store a file, edit the file or any file related task. These are all activities are achieved through the FMS. Generally the objectives of the file management system are;

- To provide the I/O support for the multi users.
- To provide a standardize set of I/O interface routines.
- Storage of data.
- To optimize performance.
- To provide I/O support for a variety of storage device types.
- To guarantee that the data in the file are valid.

File system architecture

At the lowest level, device drivers communicate directly with peripherals devices or their controls or channels. A device driver is responsible for starting I/O operations on a device and processing the completion of I/O request. The next level is the 'basic file system'. It is the interface with the environment outside of the computer system. The 'basic I/O supervisor' is responsible for all file I/O initiation and termination. 'Logical I/o' enables users and applications to access record.

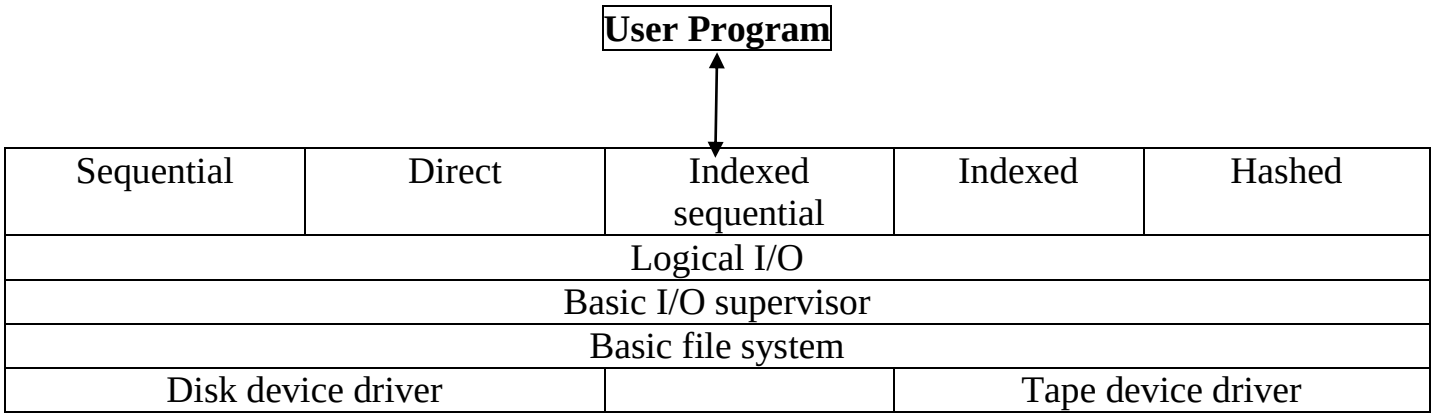


FIGURE 3: File System Architecture

Sequential file access

This method is simplest of all methods. Information in the file is processed in order, one record after the other. Magnetic tapes are supporting this type of file accessing. For example a file consisting of 100 records, the current position of read/write head is 45th record, suppose we want to read the 75th record then it accesses sequentially from 45, 46, 47, ……, 73, 74, 75. So the read /write head traverse (to visit) all the records between 45 to 75. Consider the figure 4 for better understanding.

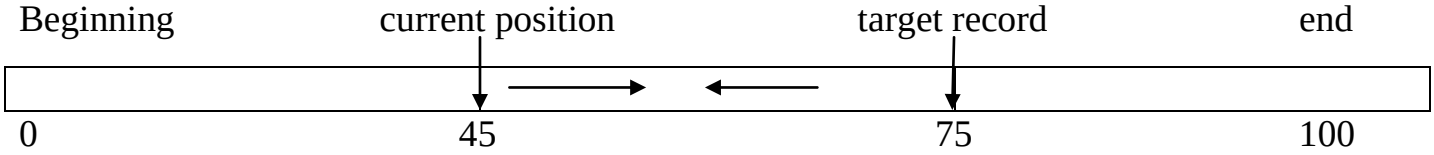


Figure 4 :Sequential file access

Direct access/Random Access/ Relative access/Arbitrary access

Direct access is also called relative access. In this method record can read/write randomly without any order. The direct access method is based on a disk model of a file, because disks allow random access to any file block. A direct file allows arbitrary blocks to be read or written. For example a disk consisting of 256 blocks, the current position of read/write head is at 95th block. The block to be read or write is 250th block. Then we can access the 250th block directly without any restrictions. Best example for direct access is C.D consisting 10 songs. At present we are listening song no. 3. Suppose we want to listen song no 9 then we can shift from song no.3 to 9 without any restriction.

Indexed sequential file.

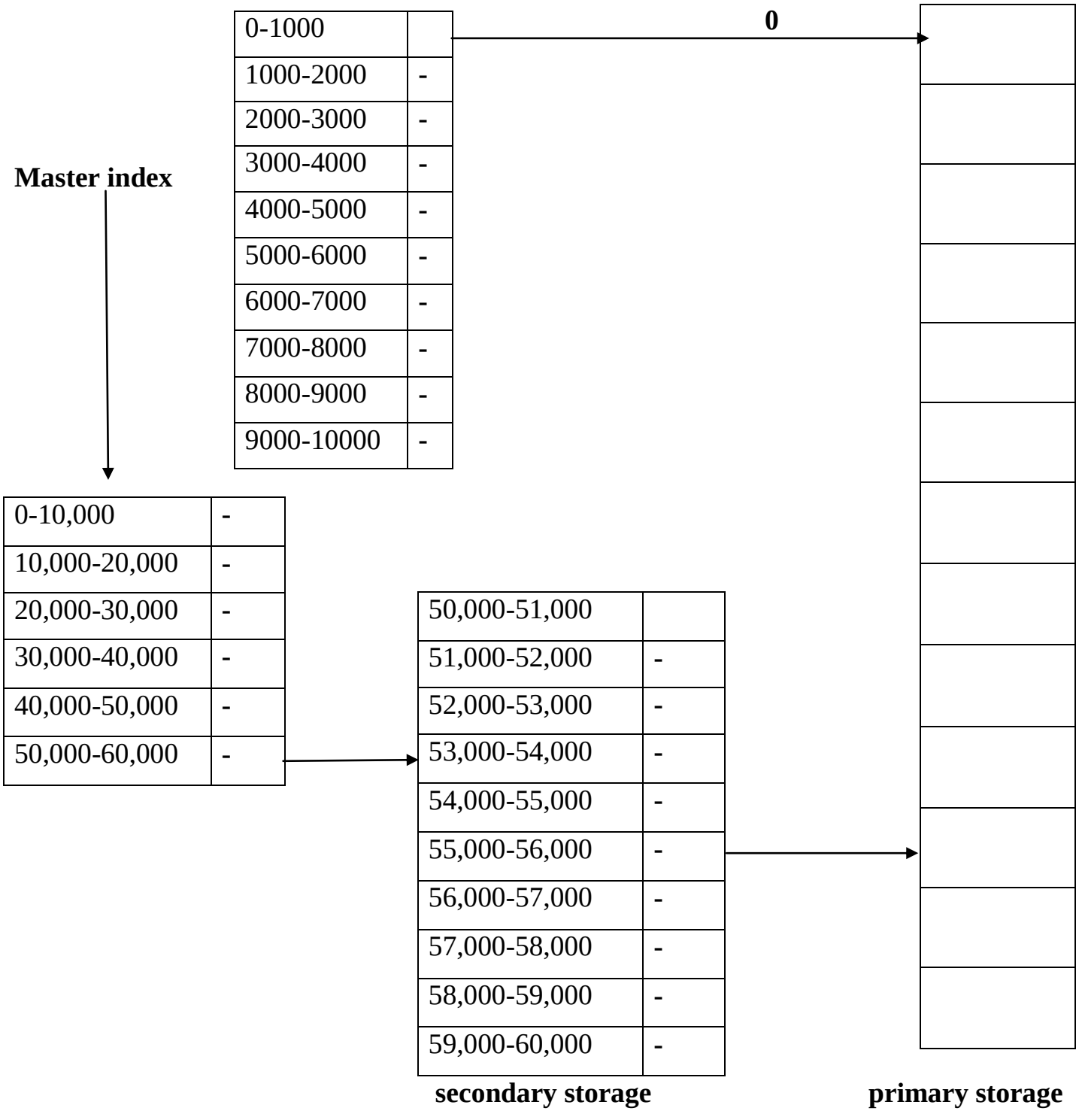


Figure 5: Index sequential file

The main disadvantage in the sequential file is, it takes more time to access a record. We can overcome this problem in this method. Records are organized in sequence based on a key field. For better understanding consider the figure 5.

Suppose a file consisting of 60,000 records, the master index divide the total records in to 6 blocks. Each block consisting of a pointer to secondary index. The secondary index is divide the 10,000 records in to 10 indexes. Each index consisting of a pointer to its original location. Each record in the index file consisting of two fields, a key field and pointer field. Suppose we want to access the $55{,}550^{th}$ record. The FMS access the index that is 50,000 to 60,000. This block consisting of a pointer, this pointer points to the 6^{th} index in the secondary index. This index points to the original location of the record from 55,000-60,000. From this, it follows the sequential method. That's why this method is said to be the indexed sequential file. Generally indexed files are used in airline reservation and payroll system.

Hashing

Hashing is the process of converting a given key into another value. A **hash function** is used to generate the new value according to a mathematical algorithm. The result of a hash function is known as a **hash value** or simply, a **hash**.

Hashing is a way of performing optimal searches and retrievals. Essentially, it increases speed, betters ease of transfer, improves retrieval, optimizes searching of data, reduces overhead. Thus, one objective of hashing is to optimize disk access and packing density.

6) FILE DIRECTORIES/FOLDERS

The directory contains information about the files including attributes, location and, ownership. Sometimes the directories consisting of subdirectories also. The directory is itself a file, owned by the operating system and accessible by various file management routines.

Directory structure

Sometimes file system consisting of millions of files, at that situation it is very hard to manage files. To manage these files grouped these files and load one group in to one partition. Each partition is called a directory. Their directory can be viewed as a symbol table that translate filenames in to their directory entries. A directory structure provides a mechanism for organizing many files in the file system.

Operations on the file directories

The operations on the directories can be performed are as follows.

Creating a directory: whenever we create a file, should make an entry in directory.

Search for directory: search the directory structure for require file.

Delete a directory: when a file is no longer needed, we want to remove it from the directory.

List a directory: we can know the list of files stored in the directory.

Rename a directory: whenever we need to change the name of the file, we can change the name of it.

Traverse the file system: we need to access every directory, and every file within a directory structure, we can traverse the file system.

7) Directory System Organization / Directory Structure

Directory structure is the mechanism used by the operating system to store directories into the computer system. Operating system follows different directory structure to organize the directories. Types of directory structures are discussed as under.

1) SINGLE LEVEL DIRECTORY STRUCTURE/SYSTEM

It is simplest of all directory structure, in this the directory system having only one directory. It conists of all files. Sometimes, it is said to be 'root directory'. For example consider the figure 6, here Root Directory contains 4 different files (a,b,c,d).

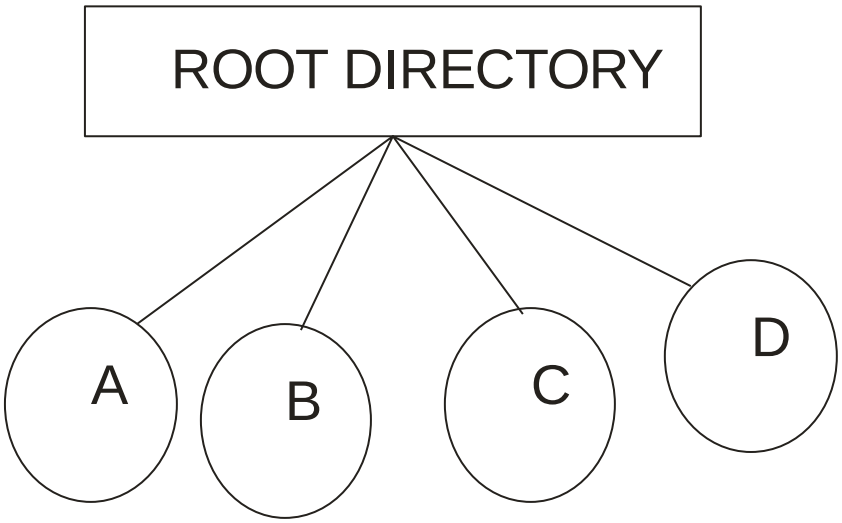


Figure 6: Single Level Directory System

The advantage of this scheme is its simplicity and the ability to locate files quickly. The problem with single level directory is different users may accidently use the same names for their files. For example, if user ONE creates a file called sample and user TWO creates the file called the same then the user TWO's file will overwrite ONE's file. That's why it is not used in multi-user system. It is used on small embedded system.

2) TWO LEVEL DIRECTORY STRUCTURE/SYSTEM

The problem in single level directory is different users may accidently use the same name for their files. To avoid this problem each user need a private directory. In this way name chosen by one user don't interfere with name chosen by another user and there is no problem caused by the same names occurring in different directories.

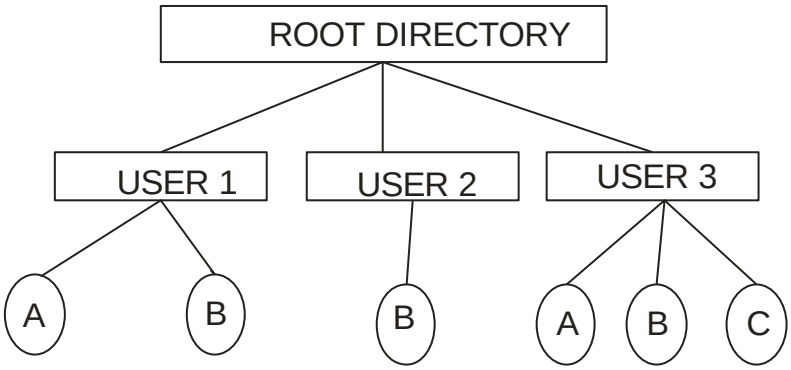


Figure 7: Two Level Directory Structure

Here, root level directory is the first level directory. It consists of entries of user directory. User 1, user2, user3 are the user levels of directories. A,b,c are files containing same name in the different directories.

3) HIERARCHICAL DIRECTORY STRUCTURE/SYSTEM

The two-level directory eliminates name conflicts among users but it is not satisfactory with the large number of files. To avoid this, create the sub directory and load the same type of files into the subdirectory. So, in this method each can have as many directories are needed. Consider the figure 8 for better understanding.

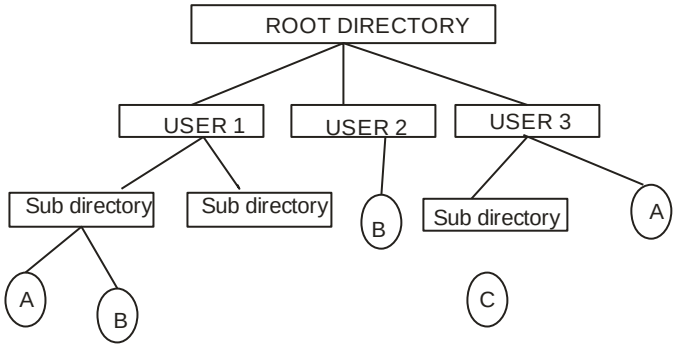


Figure 8: Hierarchical Directory System

The directory structure looks like tree structured directory, that's why it is also said to be level directory structure.

4) GENERAL GRAPH DIRECTORY STRUCTURE/SYSTEM

When we add links to an existing tree structured directory, the tree structure is destroyed, resulting in a simple graph structure. Consider the figure 9 for better understanding. The primary advantage of this structure is easy and file sharing is also possible.

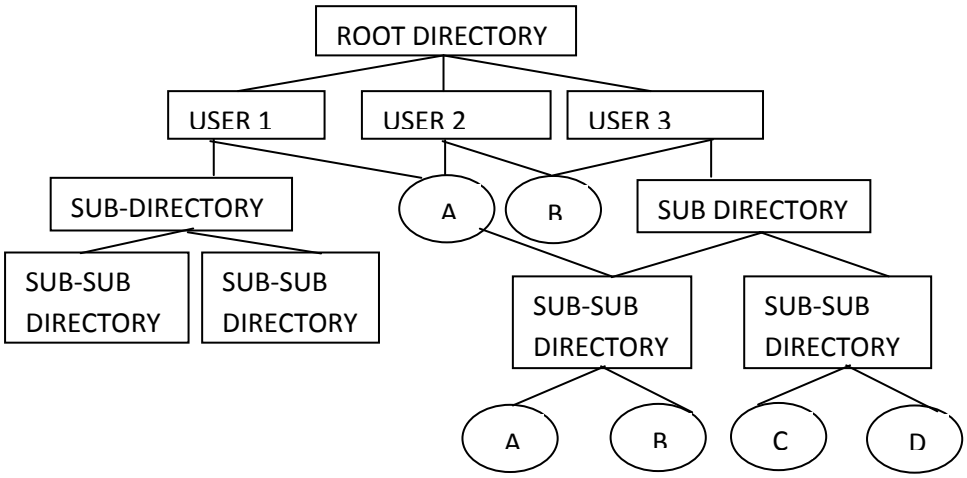


Figure 9: General Graph Directory Structure

8) FILE/DISK ALLOCATION METHODS

Files are normally stored on the disks. When file is created, storage space is allocated to it. Also, when new data is added in an existing file, file size grows and it needs extra storage space. In a similar way, storage space is released when a file is deleted or data from a file is deleted. An important function of the file system is to manage space on the secondary storage, which includes keeping track of both disk blocks allocated to files and the free blocks available for allocation.

There are two main goals, which should be fulfilled while allocating space on disk to files. These goals are as below:

1. Disk space should be utilized effectively.
2. Files should be accessed quickly.

At the time of file space allocation, system must keep track of which disk blocks go with which files. Here the disk is considered as a collection of fixed size of blocks, where size varies from system to system. Most commonly it is of 512 bytes of 1kb in size. These blocks are being numbered starting from 0 or 1 to some maximum.

But, secondary storage introduces two additional problems:

1. slows disk access time and
2. larger number of blocks to deal with

Whenever there is need to allocate storage space to a file one or more disk blocks are allocated to the file. in a similar way, whenever a file is being deleted, allocated blocks will become free for reallocation.

There are three main methods for disk allocation:

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

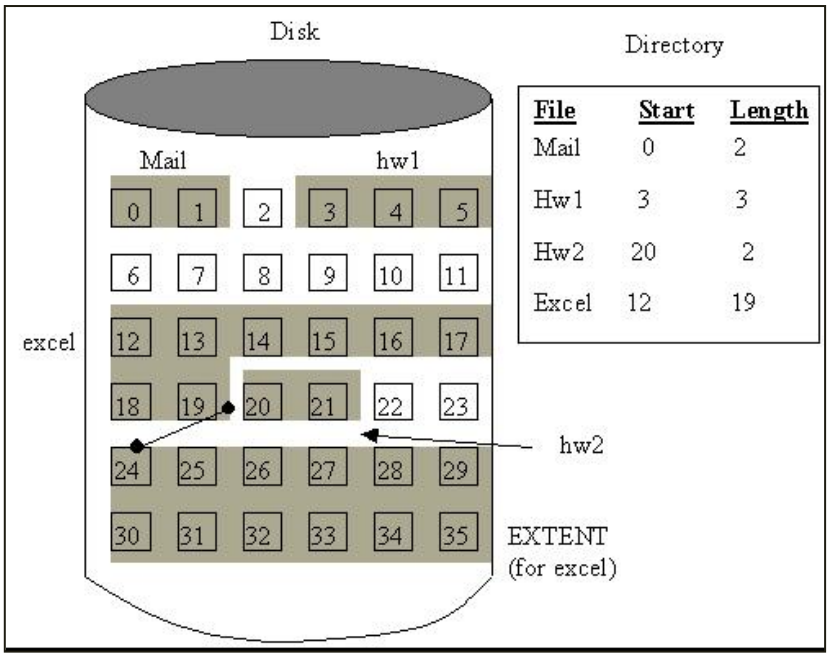
1. CONTIGUOUS ALLOCATION

In contiguous allocation, files are assigned to contiguous areas of secondary storage. A user specifies in advance the size of the area needed to hold a file to be created. If the desired amount of contiguous space is not available, the file cannot be created.

Each file occupies asset of contiguous blocks on the disk. Thus, on a disk block of 1kb, a file of 50 kb would contain 50 consecutive disk blocks. Whereas, if block size is 2kb; it will occupy 25 consecutive blocks. When a file is created, a disk is searched to find out a chunk of free memory having enough size to store a file. If such chunk is found, required memory is allocated. The directory entry contains file name, starting block number and length of a file.

File Name	Starting Block #	Length

This method is widely used on CD-ROMs. Here, all the files sizes are known in advance. Also, they will never change during subsequent uses of CD-ROM. Figure shown besides is depicting such allocation for 4 different file.



Advantages:

1. Simple to implement. Information here required is only two things; one starting block #; and second, length of file as a total numbers of blocks.
2. All successive records of a file are normally physically adjacent to each other. This increases the accessing speed of records. It means that if records are scattered through the disk it's accessing will be slower. Accessing a file which has been contiguously allocated is fairly easy.

Disadvantages:

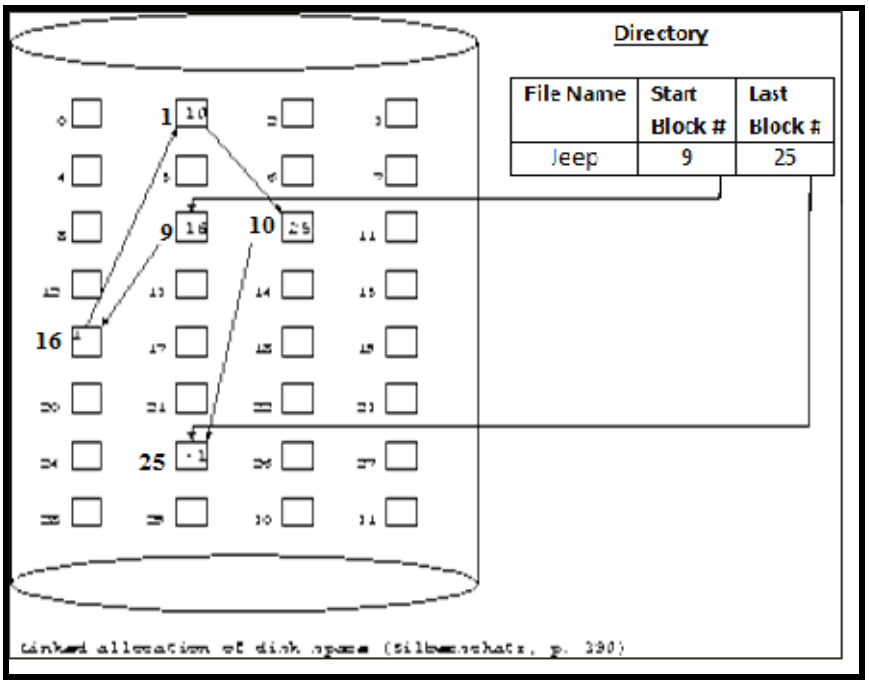
1. Finding free space for a new file is time consuming. This requires searching an entire disk until required free memory is found.
2. If size of an existing file increases, it may not be possible to accommodate such extension.

3. External fragmentation is possible. When file is deleted, its blocks are free leaving hole on the disk. With time, disk will consist of files and holes. Such hole may be too small to accommodate new files and will waste space on the disk. It is known as **external fragmentation**. Solution for this problem is defragmentation or compaction the file.

2. LINKED ALLOCATION

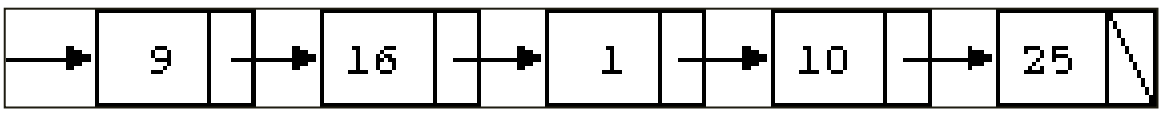
Each file is a linked list of disk blocks. Each linked block contains pointer to the next block in the list. These disk blocks may be scattered anywhere on the disk. Directory entry contains the start and last block numbers in a linked list. The directory entry structure is shown below:

File Name	Starting Block#	Last Block#



When a new file is created; a new directory entry is created. Initially it contains 'null' as both the block number. A write to the file causes free data blocks to be added to the file; such blocks are added at the end of linked list. Directory entry is updated on each such occasion. To read a file, all blocks are read by following the pointers from block to block. Following figure is depicting the linked allocation.

In the above figure ; to reach block # 10, it is required to traverse through block # 9, 16 and 1.



An important variation on the linked allocation method, called File Allocation Table (FAT), is used by the MS-DOS and older versions of Windows Operating Systems.

Advantages:

1. It does not suffer from external fragmentation.
2. Any free disk block can be allocated to a file. Such block does not need to be a consecutive block as in previous method. So, disk space can be utilized effectively.

Disadvantages:

1. File access is time consuming. It is required to access all the data blocks in a linked list to reach some particular block.
2. Random access is not possible directly.
3. Extra space is required for pointers in each data block.

3. INDEXED ALLOCATION

In linked allocation, pointers to various disk blocks are scattered on disk among various disk blocks. |Due to this reason, linked allocation cannot support efficient direct access. Indexed allocation solves this problem.

It brings all the pointers together into one location; the Index Block. Each file contains its own index block. An index block is an array of 'disk block addresses'. The 'i^{th}' entry in the index block points to the 'i^{th}' block of the file. Directory entry contains file name and the index block #. This looks as shown below:

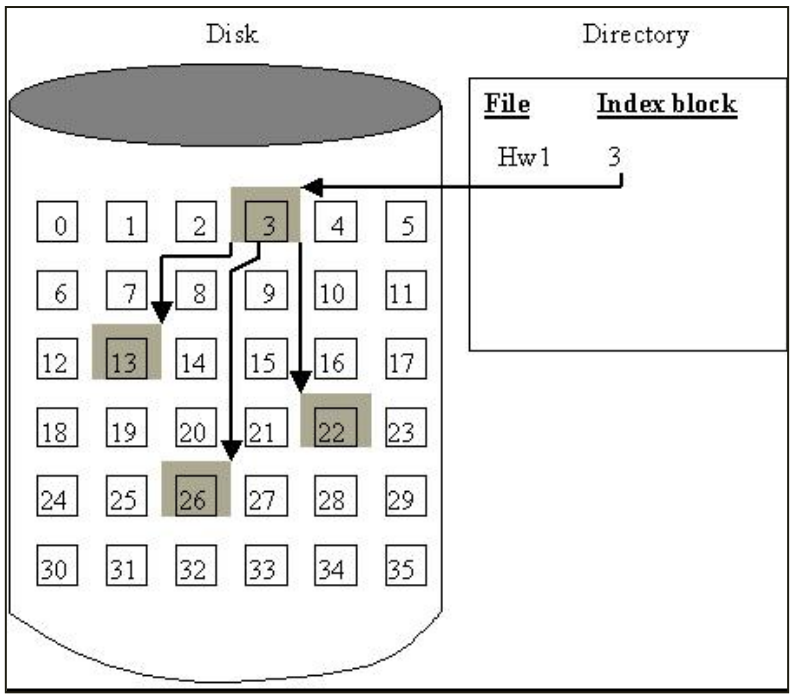
File Name	Index Block#

When new file is created, a new directory entry is created. Initially all pointers in index block are set null. When the 'i^{th}' block is first written, a free disk block is allocated and its address is put in the 'i^{th}' entry in the index block.

The following figure depicts the indexed allocation for the file 'Hw1'. The directory entry for the index block is done as block #3 for this file. This index block entry is composed of all other block # entries which are having the data contents of the file Hw1.

The index entry consists of all the block entries which consists of the data contents.

13	26	22



Advantages:

1. It does not suffer from external fragmentation.
2. Direct access is efficient.

Disadvantages:

1. It suffers from wasted space. Index block may be partially filled. This wastes remaining memory space of an index block. For example, if file contains two data blocks, then index block will have only two entries to point to these two data blocks, remaining entire index block will be wasted,
2. Maximum allowable file size depends on the size of an index block.

Solutions:

In this allocation method the main problem is the size of index block. If it is too large, it may waste space. If it is too small it cannot accommodate enough pointers for a large file. Following are the solutions:

1. Linked Scheme

An index block is normally of one disk block size. For larger files more than one index block can be used by linking them together by a linked list.

2. Multilevel Index / Hierarchical Indexing

Two or more level of index blocks is used here. First level index blocks point to a set of second level index blocks. Second level index blocks point to disk blocks containing the data. For larger and larger file, levels can be increased.

3. Combined Scheme

In this solution, partial entries from indirect block point to direct block containing file data. Partial entries point to indirect blocks, which are index blocks. They do not contain file data, but they contain address of disk blocks (pointers) that do contain file data. UNIX Os is using such type of combined scheme.

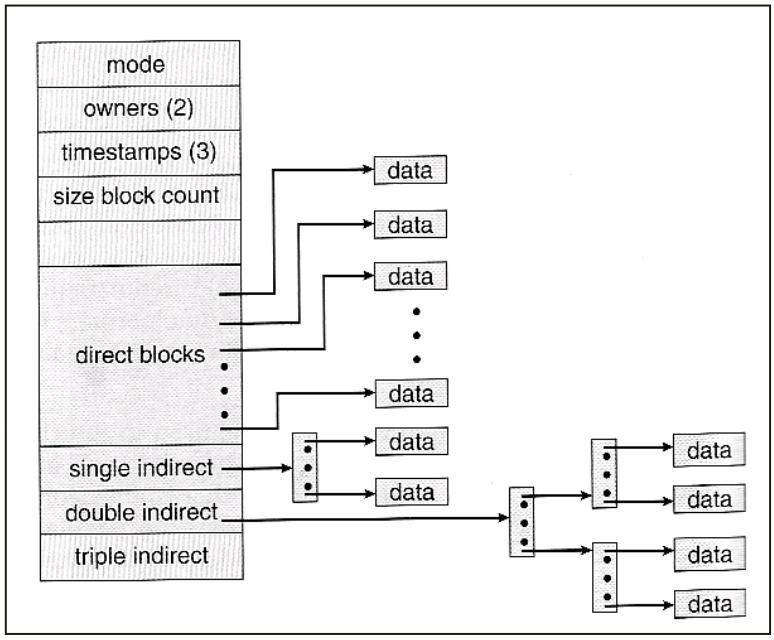


Figure shown besides is an example of such combined scheme which is implemented in the UNIX operating system for storing the I node.

9) FILE PROTECTION

File protection means protecting a file from unauthorized and unauthenticated access or user. File is protected for following purposes;

1) Preventing unauthorized access
2) Preventing accidental erasing of data

3) Enforce appropriate permissions
4) Prevent hackers to access the file
5) Prevents programs from replacing critical system files
6) Programs must not overwrite wrong file
7) Protecting from virus or malicious program

There are many ways to protect file which are discussed as under.

1) **Logical File Protection:** Logical file protection is provided by the operating system, which can designate files as read only. This allows both regular (read/write) and read only files to be stored on the same disk volume.
2) **Hiding File:** Files can also be made as hidden files, which make them invisible to most software programs.
3) **Password Protection:** File can access through pre-defined password. Many software such as PDF, MS-Office provide a facility to lock a file through password and unlock a file by entering the correct password.
4) **Anti-Virus Protection:** File can be protected by third party anti-virus software such as QuickHeal, Bit Defender, Norton, McAfee etc. These softwares provide a facility to protect file from unauthorized access by providing a password facility.
5) **Encryption & Decryption:** Encryption means converting file information into secret information or data using a sophisticated encryption algorithm. When, file data is to be read by authorized person, then, decryption is used to convert encrypted information back to original form using sophisticated decryption algorithm.

3.1 Introduction to Linux Operating System

- Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.
- Linux is a UNIX-base operating system. Its original creator was a Finnish student name Linus Torvalds, although being 'open source' it has change a great deal since its original conception.
- It belongs to nobody, and is free to download and use. Any changes to it are open for all to adopt, and as a result it has developed into a very powerful OS that is rapidly gaining in popularity worldwide, particularly among those seeking an alternative to Windows.

- In 1991, hardware was expanding rapidly, and DOS was the king of operating systems. Software development was slower, and Macs, while better, were also much pricier than PCs. UNIX was growing, but at that time in its history the source code was jealously guarded and expensive to use.
- Linus Torvalds was a Helsinki university student who liked playing around with software and computers, and in 1991 he announced the creation of a new core operating system that he had named Linux.
- It is now one of the most used systems for the PC, and is particularly suitable for businesses with small IT budgets.
- Linux is free to use and install, and is more reliable than almost all other systems, running for many months and even years without a reboot being necessary.

- Version of linux:
  - Android
  - Arch Linux
  - Debian GNU/Linux
  - Gentoo Linux
  - Kubuntu
  - Mandriva Linux
  - PC LinuxOS
  - Linux for playstation 2
  - Red Hat Linux
  - Sabayon Linux
  - Slackware
  - SUSE Linux
  - Ubuntu

Distribution	Description
Red Hat Linux	Split into Fedora Core and Red Hat Enterprise Linux. The last official release of the unsplit distribution was Red Hat Linux 9 in March 2003.
CentOS	Community-supported Linux distribution designed as an OpenSource version of RHEL and well suited for servers.
Fedora	Community-supported Linux distribution sponsored by Red Hat. It usually features cutting-edge Linux technologies.
openSUSE	A community-developed Linux distribution, sponsored by SUSE. It maintains a strict policy of ensuring all code in the standard installs will be from FOSS solutions, including Linux kernel Modules. SUSE's enterprise Linux products are all based on the codebase that comes out of the openSUSE project.
Mandrake Linux	The first release was based on Red Hat Linux (version 5.1) and KDE 1 in July 1998. It had since moved away from Red Hat's distribution and became a completely separate distribution. The name was changed to Mandriva, which included a number of original tools, mostly to ease system configuration. Mandriva Linux was the brainchild of Gaël Duval, who wanted to focus on ease of use for new users.

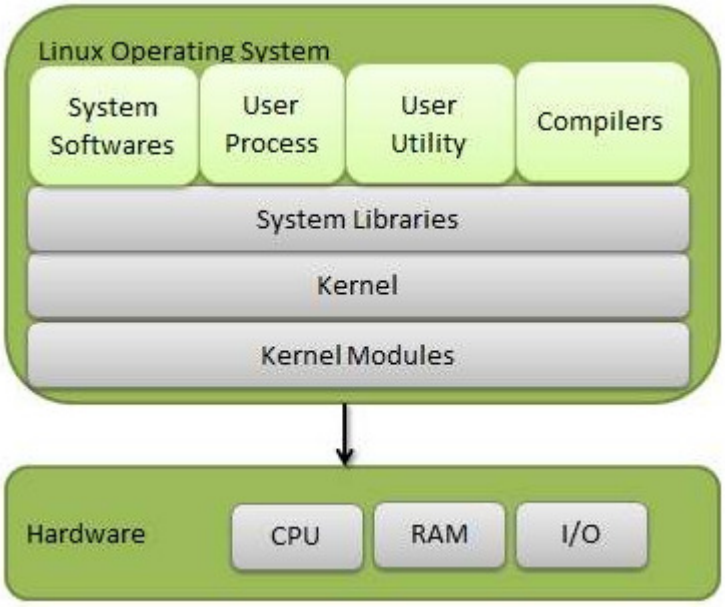
URL	Site Description
redhat.com	Red Hat Linux
fedoraproject.org	Fedora Linux
centos.org	Centos Linux
opensuse.com	openSUSE Linux
debian.org	Debian Linux
ubuntu.com	Ubuntu Linux
mepis.org	Mepis Linux
gentoo.org	Gentoo Linux
turbolinux.com	Turbo Linux
knoppix.org	Knoppix Linux
linuxiso.com	CD-ROM ISO images of Linux distributions
distrowatch.com	Detailed information about Linux distributions
kernel.org	Linux kernel

TABLE 1-1 Linux Distribution and Kernel Sites

3.2 Components of Linux System

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** – System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** − Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.

- **Open Source** – Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** – Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** – Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

3.3 Comparison of Windows and Linux.

Windows	Linux
Windows uses different data drives like C: D: E to stored files and folders.	Unix/Linux uses a tree like a hierarchical file system.
NT needs 12 MB RAM	Linux needs 2MB RAM
NT needs 70 MB at least.	Linux needs at least 15 MB disk space
Windows has different drives like C: D: E	There are no drives in Linux
There are 4 types of user account types 1) Administrator, 2) Standard, 3) Child, 4) Guest	There are 3 types of user account types 1) Regular, 2) Root and 3) Service Account
Administrator user has all	Root user is the super user and

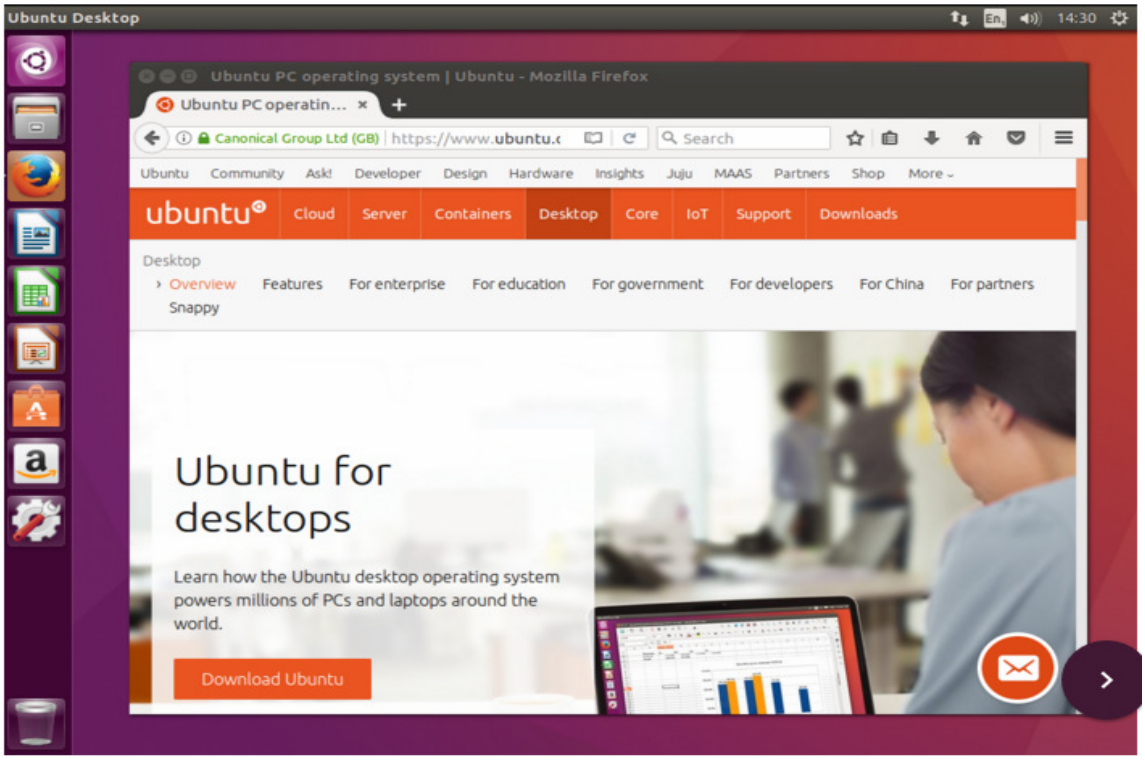
administrative privileges of computers.	has all administrative privileges.
In Windows, you cannot have 2 files with the same name in the same folder	Linux file naming convention is case sensitive. Thus, sample and SAMPLE are 2 different files in Linux/Unix operating system.
In windows, My Documents is default home directory.	For every user /home/username directory is created which is called his home directory.
Window is not command line interface	Linux is command line interface.
No access This is not possible with windows as	**Full access** Linux belongs to the GNU Public License ensures that users can access (and alter) the code to the very kernel that serves as the foundation of the Linux operating system.
Licensing restriction You are bound to the number of licenses you purchase, so if you purchase 10 licenses, you can legally install that operating system (or application) on only 10 machines.	**Licensing freedom** you can download a single copy of a Linux distribution (or application) and install it on as many machines

Unit 4. Linux Administration

4.1. Installing Linux
4.2. Installation of Open Source Software
4.3. Maintaining User Accounts
4.4. System Config Services (Package)

4.1 Installing Linux

The Ubuntu desktop is easy to use, easy to install and includes everything you need to run your organization, school, home or enterprise. It's also open source, secure, accessible and free to download.

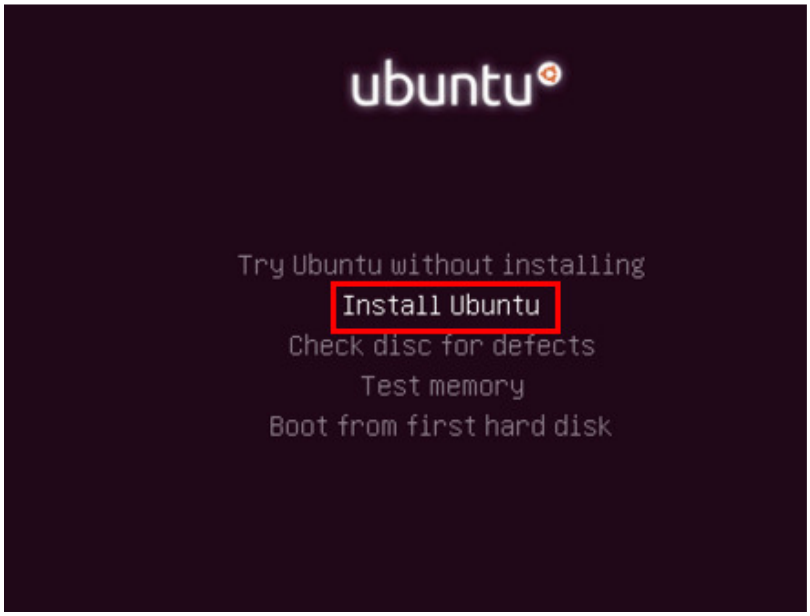


- Make sure you have a recent backup of your data. While it's unlikely that anything will go wrong, you can never be too prepared.

This is a step-by-step installation procedure for Linux, specifically Ubuntu 16.04.

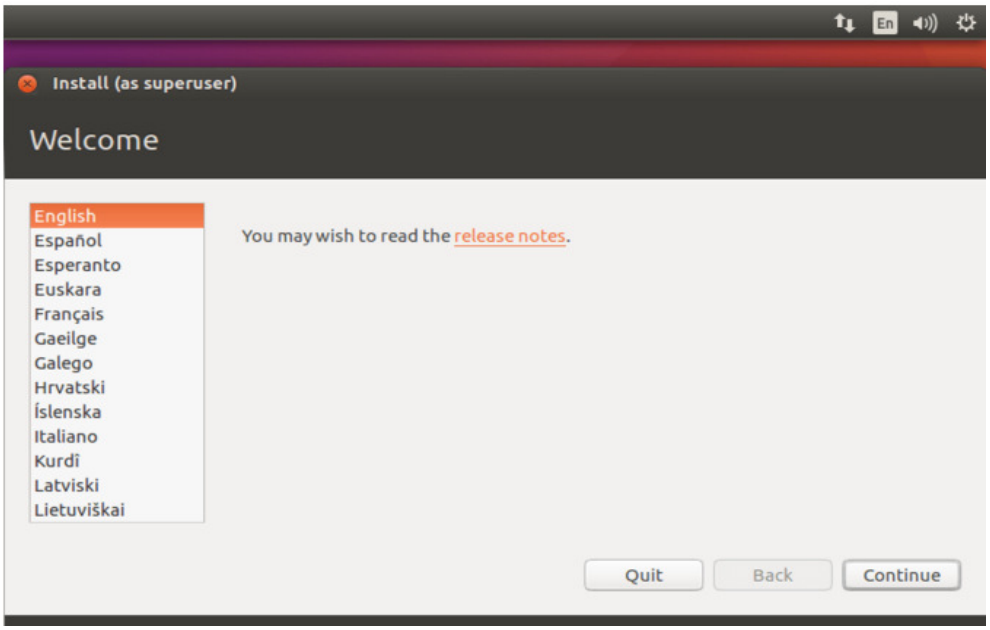
Step1 – Preparing Installation

For installing the Ubuntu 16.04, Select Install Ubuntu.

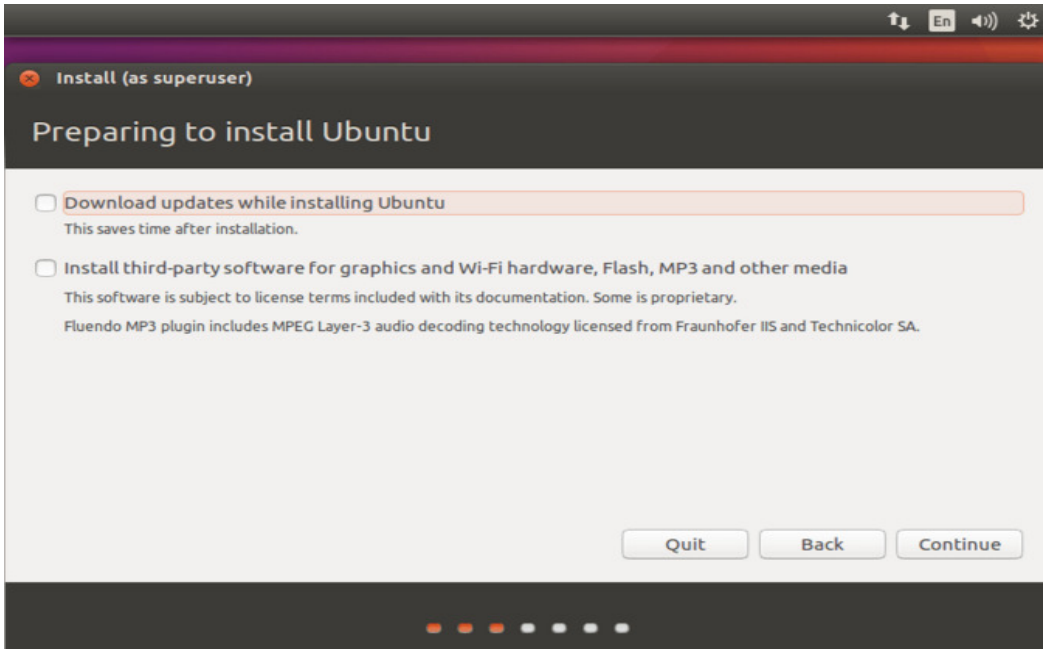


Step 2 – Welcome Screen

Press Enter to get a language screen and then select the language of your choice and click to continue



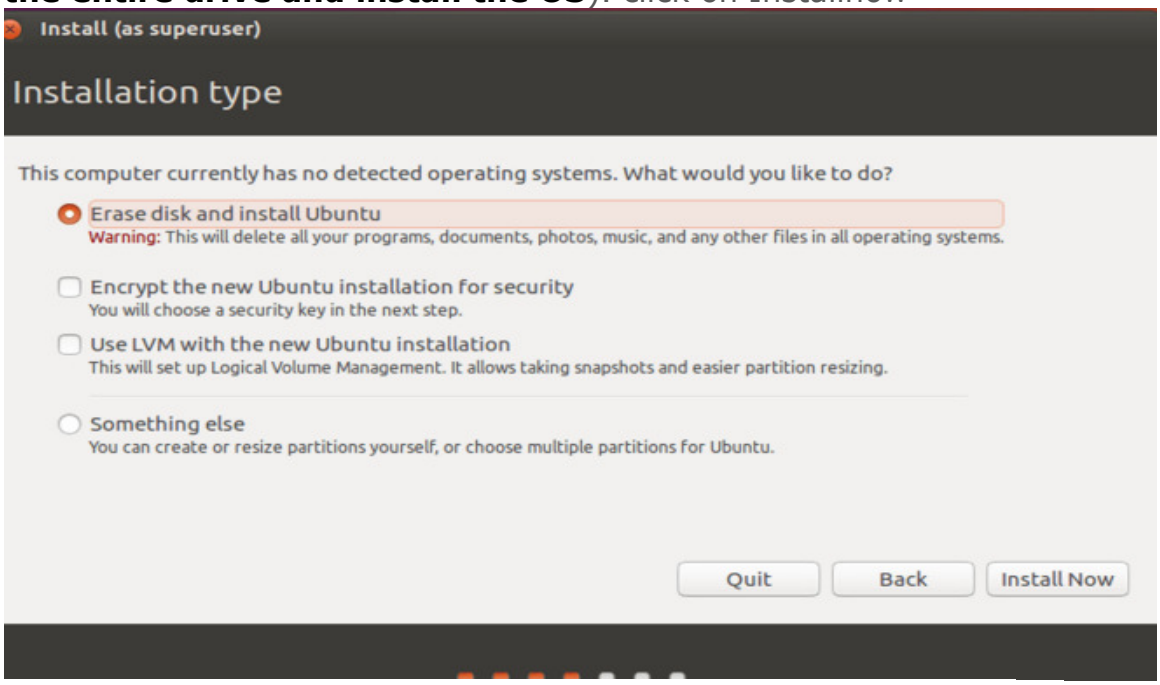
You can either choose to **install updates** and **other third-party software** while installing Ubuntu 16.04.



Step3 – Installation Type

We have only two option in the installation type.Please chose any one of the methods.

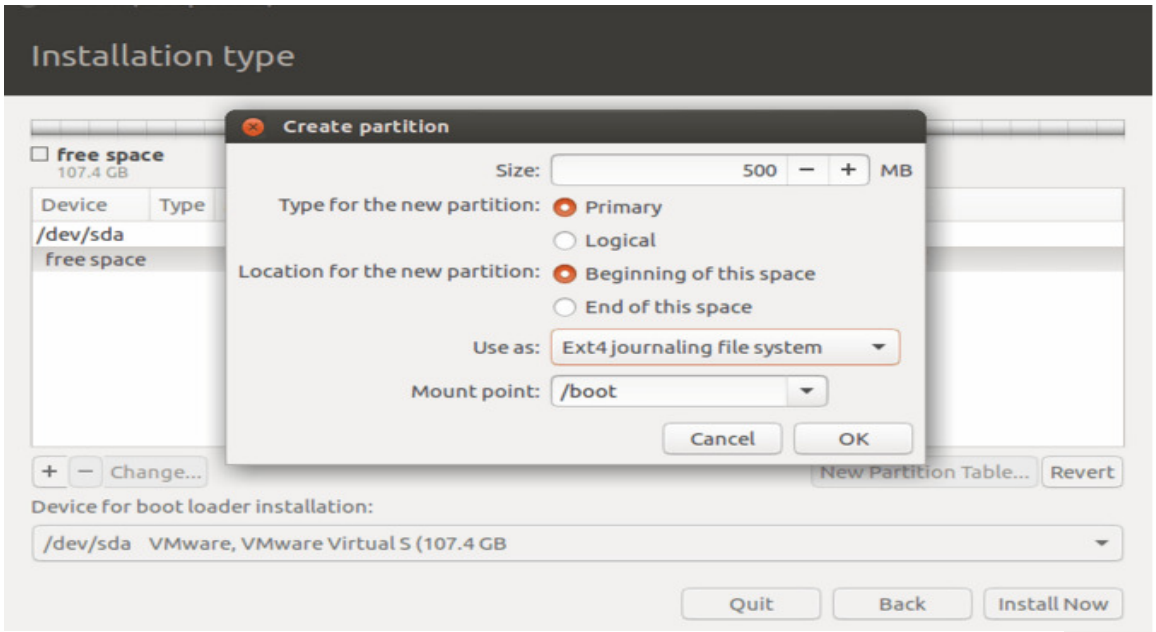
1. Erase full disk : Erase disk and install Ubuntu (i.e. **it will format the entire drive and install the OS**). click on Installnow



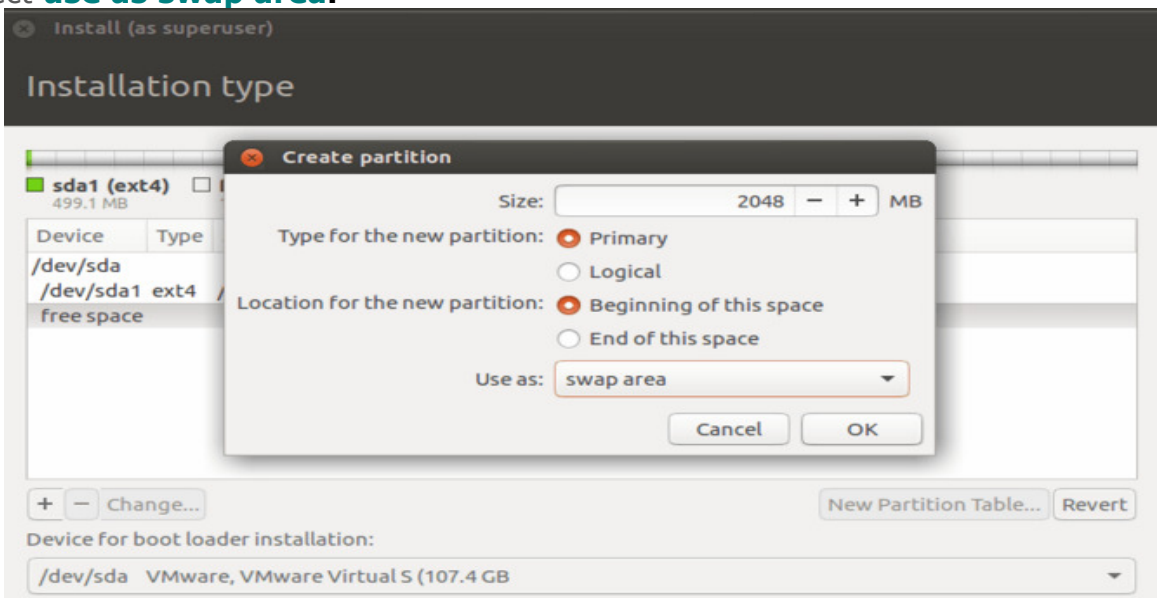
Once you clicked on Install Now, the installer will ask you to confirm the auto partitioning. Click on continue.

2. **Boot Partition : Something else** (i.e. **you can manually create the partition and install Ubuntu on your selected partition**), use this advanced mode if you are comfortable in partitioning your drives manually. Click on continue.

   Select free space and click on the **+ sign** at the bottom to create partitions. Following shows for /boot partition.

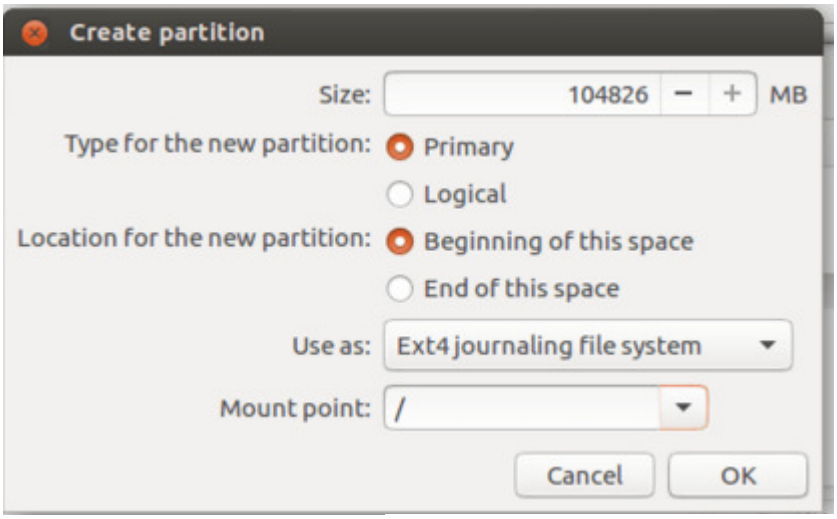


Step4 – Swap Following screen show for the swap partition, it is important to select **use as swap area.**

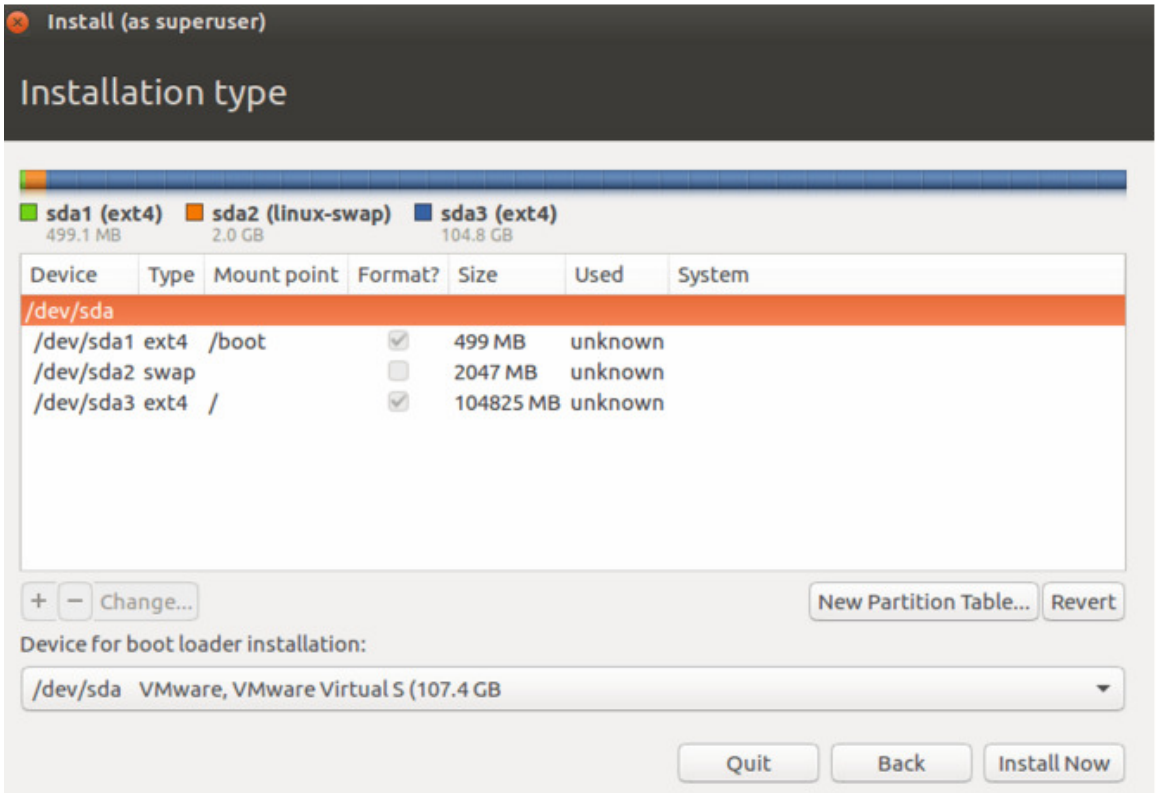


Step5 – root partition

Following is for / (root) partition.

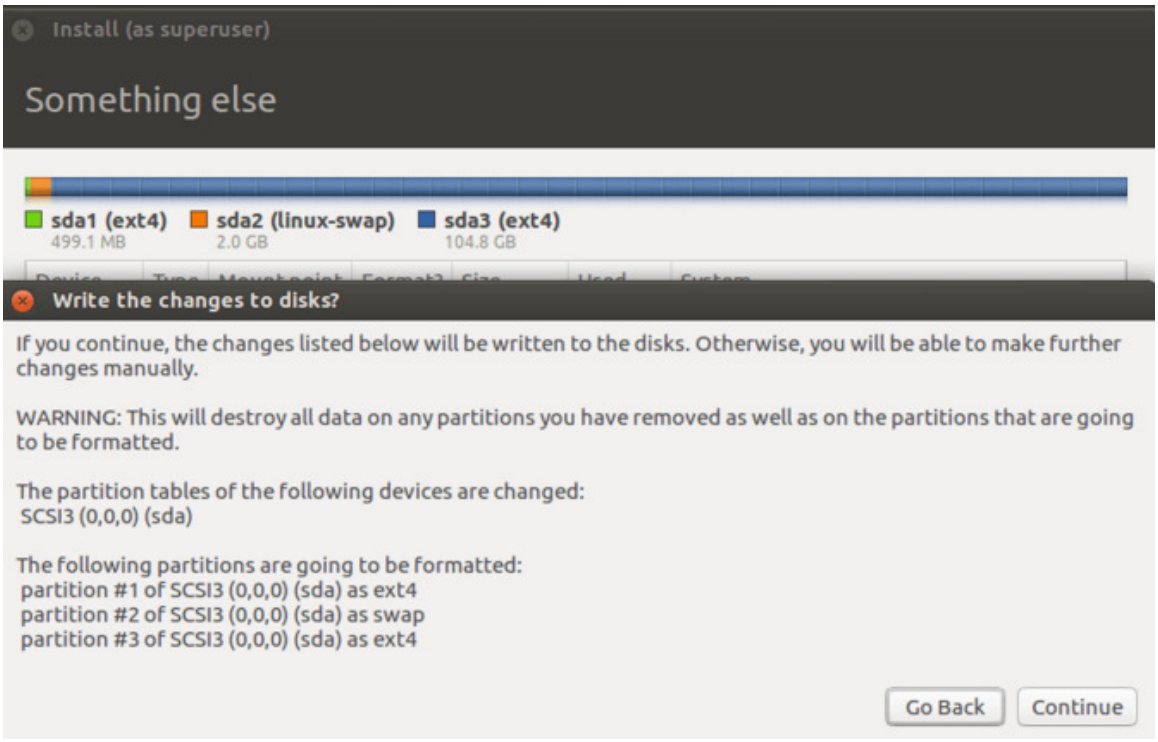


Step6 – Partition List : Review your partition layout and click on install now.



Step7 – Formatting Partitions

Write the changes to disk by clicking on continue.

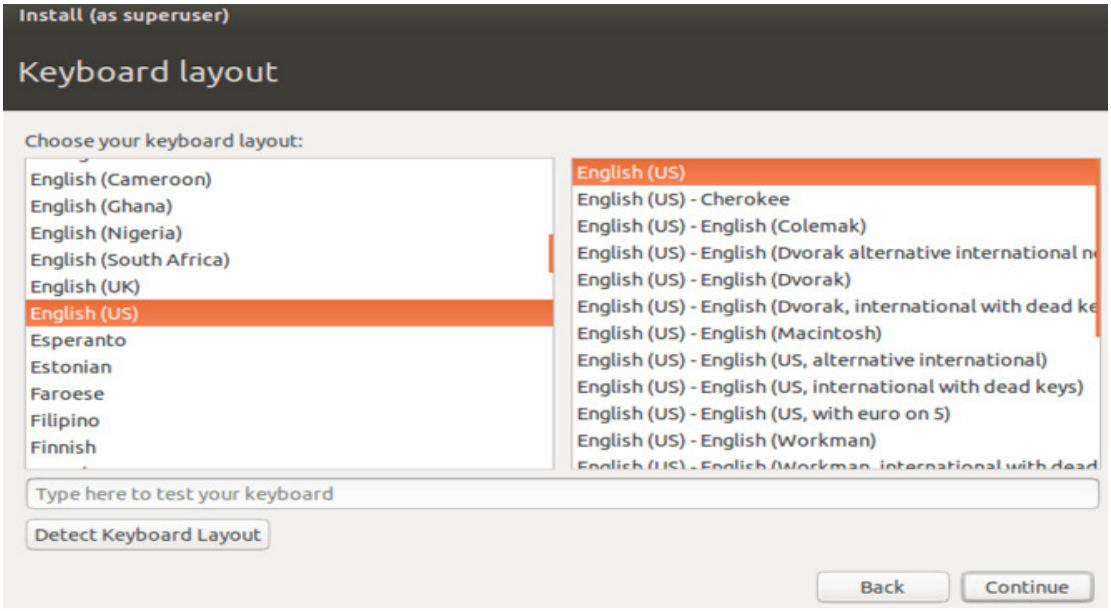


Step8 – Select Location

Select your location Press continue.

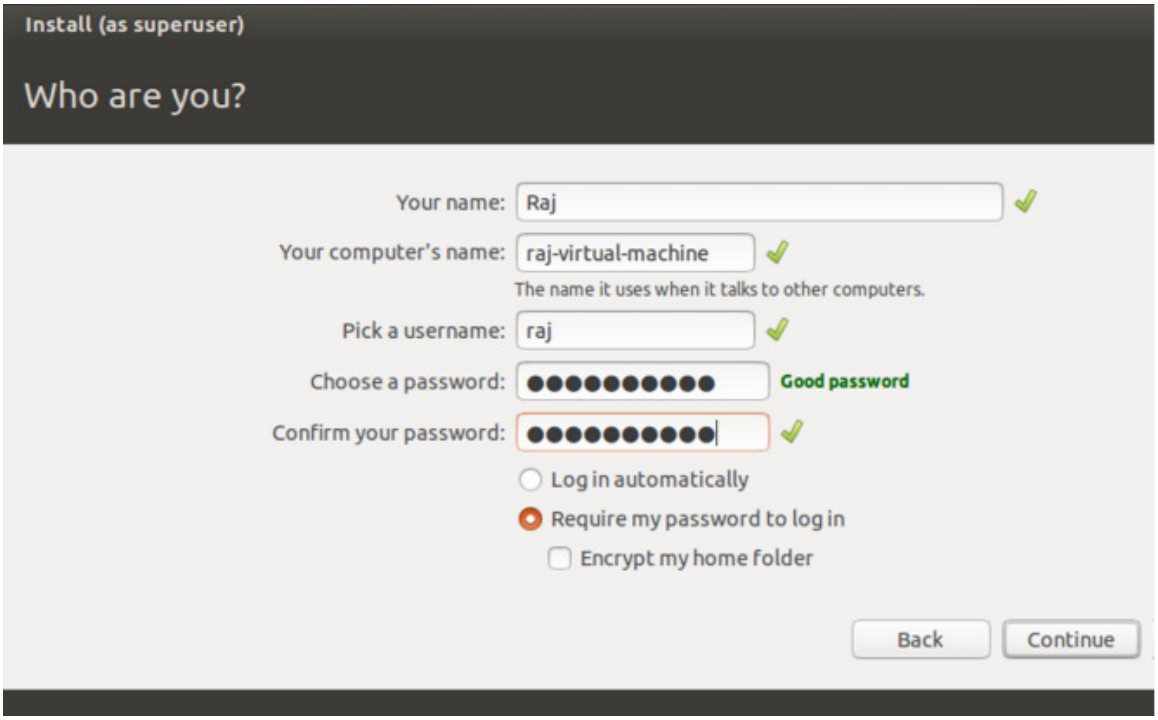
Step9 – Keyboard Layout

Select your keyboard layout. If you are not sure, use the `**Detect Keyboard Layout**` option. You can also test your selection by typing in the test text box.



Step10 – User

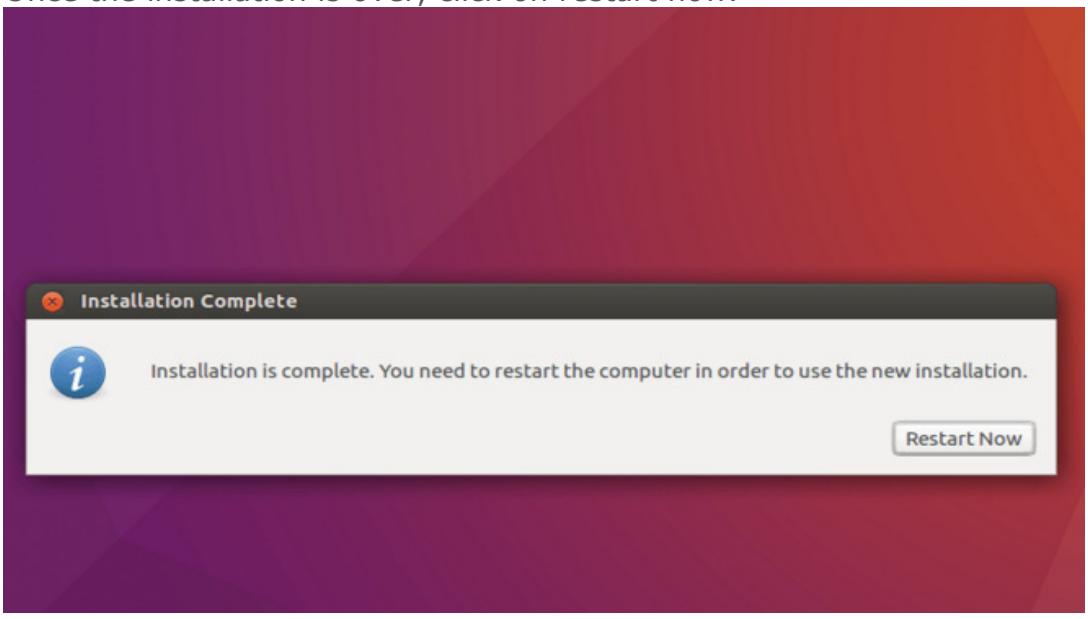
click on continue.



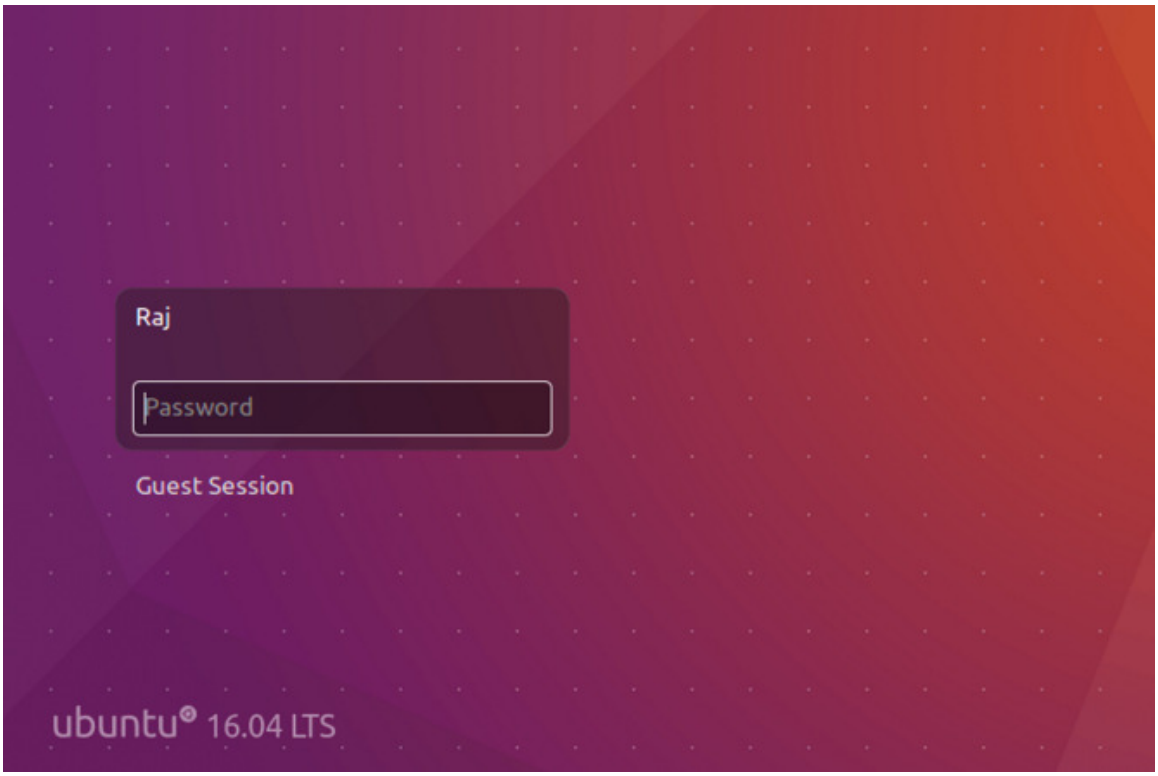
Step11 – Installing

Below screenshot shows installing Ubuntu 16.04.

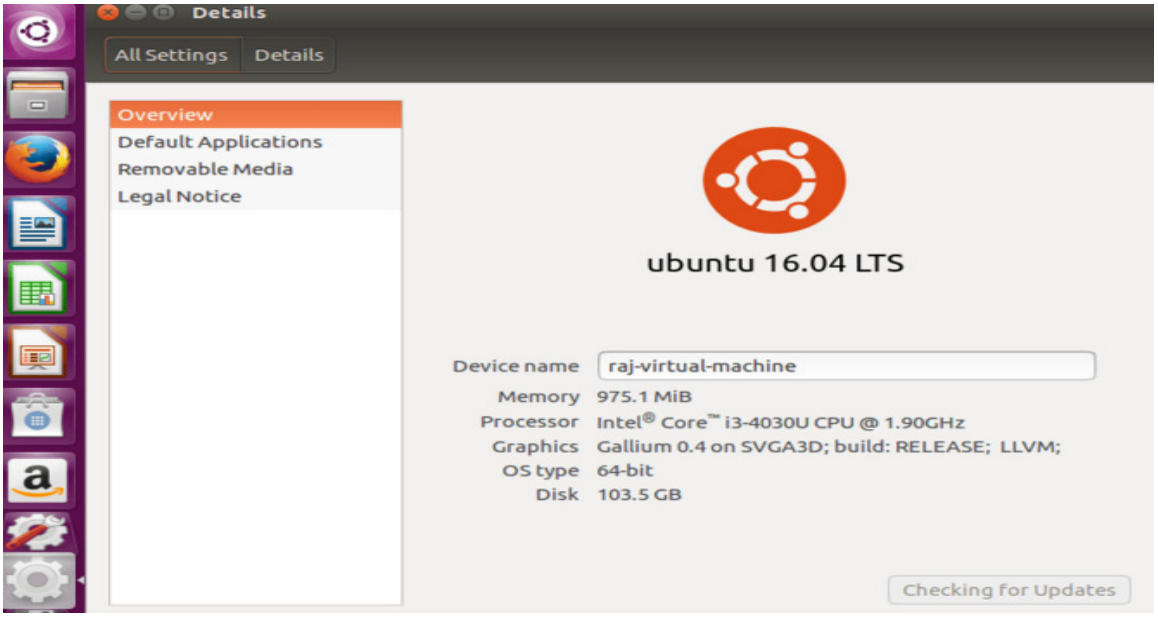
Step12 – Restart After the installation
Once the installation is over, click on restart now.



Once your machine is restarted, you will get a login window. Login with username and password that you created earlier.



Install Ubuntu 16.04 – Desktop Screen



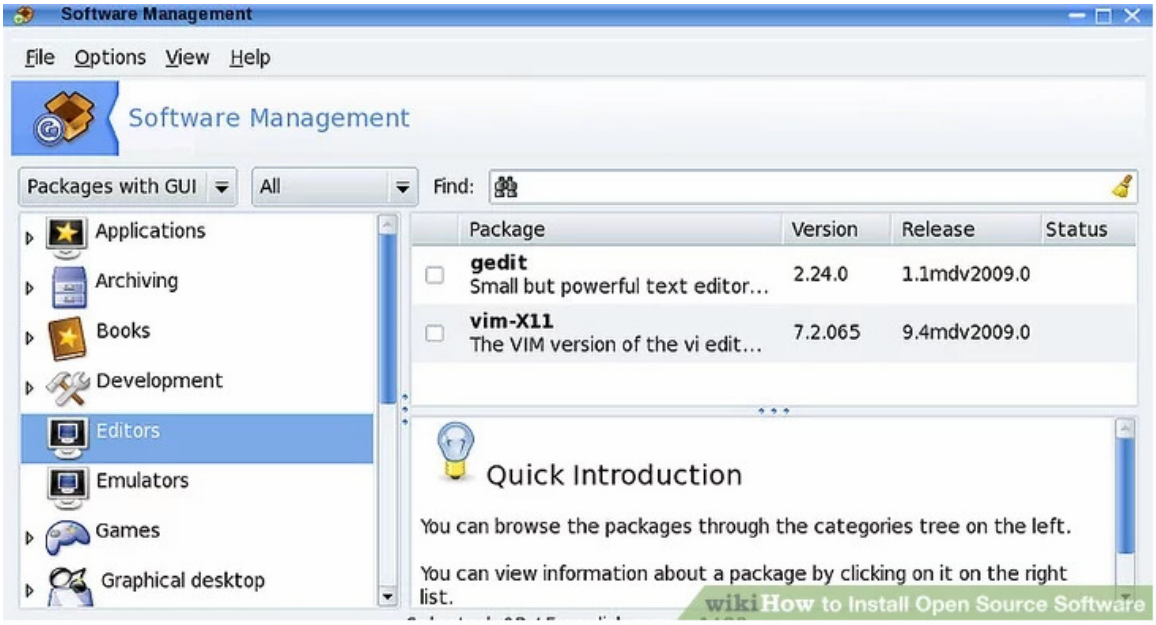
Ubuntu is now ready for you to try it out!! Use, Share and Enjoy.

4.2 Installation of Open Source Software

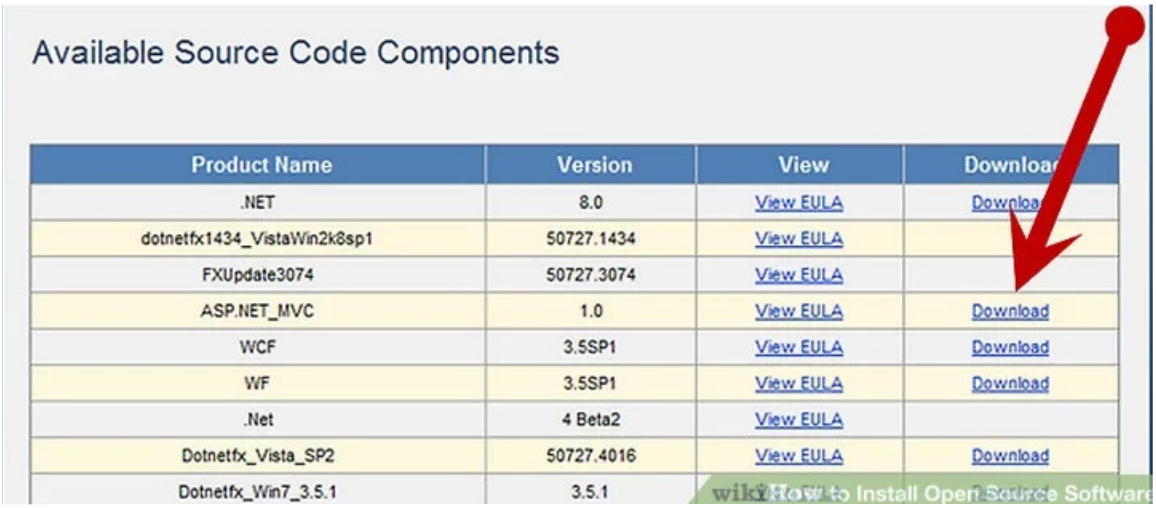
Once you have decided to migrate to open source software, you will need to do some basic installing.

Installing open source software depends on your operating system. This is a how-to compilation for multiple operating systems; read the appropriate section for your OS.

Linux/Unix/Unix-Like Systems



For most such systems, you can probably use the OSs package manager to install a pre-built binary package. This is always the recommended method.



- Alternatively, you could follow these steps:
  - Download and uncompress the source code.
  - In the terminal, move into the extracted directory.
  - Run "./configure" to configure the software.
  - Run "make" to compile the software.
  - Run "make install" to install the software.

4.3. Maintaining User Accounts

- **Linux user**
- A user or account of a system is uniquely identified by a numerical number called the UID (unique identification number).
- There are two types of users –
  1. the root or super user.
  2. Normal users.
- A root or super user can access all the files, while the normal user has limited access to files.
- A super user can add, delete and modify a user account. The full account information is stored in the */etc/passwd* file and a hash password is stored in the file */etc/shadow*. Some operations on a user account are discussed below.
- **Creating a user with a default setting:** A user can be added by running the *useradd* command at the command prompt. After creating the user, set a password using the *passwd* utility.
  - The system automatically assigns a UID, creates the **home directory (*/home/<username>*) and sets the default shell to */bin/bash*.**

    - The *useradd* command creates a user private group whenever a new user is added to the system and names the group after the user.
- **Locking and unlocking a user:** A super user can lock and unlock a user account.
    - To lock an account, one needs to invoke *passwd* with the *-l* option.
    - To unlock an account, one needs to invoke *passwd* with the *-u* option.
- **Changing a user name:** The *–l* option with the *usermod* command changes the login (user)
- **Removing a user:** Combining *userdel* with the *–r* option drop a user and the home directory associated with that user.
- **Linux group**
  Linux group is a mechanism to organize a collection of users. Like the user ID, each group is also associated with a unique ID called the GID (group ID).
- There are two types of groups –
  1. **a primary group**
  2. **a supplementary group**.
- Each user is a member of a primary group and of zero or 'more than zero' supplementary groups.
- The **group information** is stored in ***/etc/group*** and the respective **passwords** are stored in **the** ***/etc/gshadow*** file.
- Some operations such as creating, deleting and modifying on a group are below.
- **Creating a group with default settings:** To add a new group with default settings, run the ***groupadd*** command as a root user.
- If you wish to add a password, then type ***gpasswd*** with the group name.

- **Creating a group with a specified GID:** To explicitly specify the GID of a group, execute **the** ***groupadd*** command with the ***–g*** option.

- **Removing group password:** To remove a group password, run ***gpasswd –r*** with the relevant group name

- **Changing the group's name:** To change the group's name, run the ***groupmod*** command with the ***-n*** option as a super user

- **Changing the group's GID:** To change the GID of a group, run the ***groupmod*** command with ***–g***

- **Deleting a group:** Before deleting a primary group, delete the users of that primary group. To delete a group, run the ***groupdel*** command with the group name

4.4 System Config Services (Package)

- **Name:** system-config-services - Service Configuration Utility
- **Synopsis:** system-config-services
- **Description :** This is a graphical tool for enabling and disabling services (including xinetd services). Functionality to start, stop, and restart services is also included.
- **Options :**None
- **Files:**/usr/bin/system-config-services
    - /usr/share/system-config-services/*
- **To run this program simply type:** system-config-services
- **Bug :**

- Some services will not start or stop properly if started anywhere but the console (system-config-services will appear to hang in these instances). This is not a bug in system-config-services, but in the individual services.

- Some services are incredibly hard to detect if they are running or not. While there are workarounds present to deal with these, it can't be guaranteed that they're detected properly. Please file bugs against the system-config-services component at http://bugzilla.redhat.com if you encounter such services.

Some configuration files run a set of commands upon startup. A common convention is for such files to have "rc" in their name, typically using the name of the program then an "(.)rc" suffix e.g. ".xinitrc", ".vimrc", ".bashrc", "xsane.rc". S

There are various methods for managing access to system services:
a) /etc/init.d/service
b) rcconf
c) update-rc.d etc

DEVICE MANAGEMENT

Devices are all pieces of equipment for a computer that perform Input / Output (*short:* I/O) operations but that the computer does not necessarily need to work.

In some way all parts of a computer perform I/O operations, but without CPU or memory a computer would not work at all. Therefore CPU and memory are no devices whereas hard disk drives, mouse, keyboard, scanner, sound card and so on all are devices.

Devices are sometimes referred to as *peripheral devices* to emphasize the point that they are "additional" to the core hardware system. In this section the main effort is put on storage devices, that are devices that store data permanently.

Device Management is needed for offering a uniform and consistent approach to all I/O operations. There are considerable differences between all the system's devices; their speed, how data are transferred and represented, how to prevent and detect errors and how they are handled.

DEVICE MANAGEMENT FUNCTIONS

The basic functions of device management are as mentioned below:

1. Keeping track of the status of all devices, which requires special mechanism. One commonly used mechanism is to have a database such as Unit Control Block (UCB) associated with each device.
2. Deciding on policy to determine who gets a device, for how long and when. A wide range of techniques is available for implementing these policies. For example, high device utilization attempts to match the non-uniform requests by processes to the uniform speed of many I/O devices. There are 3 basic techniques for implementing the policies of device management:
   a. **Dedicated**: A technique whereby a device is assigned to a single process.
   b. **Shared**: A technique whereby a device is being shared by many processes.
   c. **Virtual**: A technique where one physical device is simulated on another physical device.
3. Allocation- physical assigning device to a process. Likewise the corresponding control units and channels must be assigned.
4. De-allocation policy and techniques. De-allocation may be done on either a process or a job level. On a job level, a device is assigned for as long as the job exists in the system. On process level, a device is assigned for as long as the process needs it.

DEVICE CHARACTERISTICS

Devices which are used to perform I/O can be grouped into three categories:

- **Human readable:** Suitable for communication with the computer user. Examples include printers and terminals, keyboard, and perhaps other devices such as a mouse.
- **Machine readable:** Suitable for communicating with electronic equipment. Examples are disk drives, USB keys, sensors, controllers and actuators.
- **Communication:** Suitable for communicating with remote devices. Examples are digital line drivers and modems.

Devices can be categorized using following criteria,

- **Data rate:** There may be differences of several orders of magnitude between the data transfer rates.
- **Application:** The user to which a device is put has an influence on the software and policies in the operating system and supporting utilities. For example a disk used for files requires the support of file management software. A disk used as s backing store for pages in a virtual memory scheme depends on the use of virtual memory hardware and software. These applications have an impact on disk

scheduling algorithms. For example a terminal may be used by an ordinary user or a system administrator.

- **Complexity of control:** A printer requires a relatively simple control interface. A disk much more complex. The effect of these differences on the operating system is filtered to some extent by the complexity of the I/O module that controls the device.
- **Unit of transfer:** Data may be transferred as a stream of bytes or characters or in large blocks.
- **Data representation:** Different data encoding schemes are used by different devices, including differences in character code and parity conventions.
- **Error condition:** The nature of errors, the way in which they are reported, their consequences, and the available range of responses differ widely from one device to another.

DISK SPACE MANAGEMENT

- ### DISK FORMATTING

A new magnetic disk is a blank slate. It is just a platter of a magnetic recording material. Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting, or physical formatting. Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a header, a data area, and a trailer. The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC). When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area. When the sector is read, the ECC-is recalculated and is compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad. The ECC is an error-correcting code because it contains enough information that, if only a few bits of data have been corrupted, the controller can identify which bits have changed and can calculate what their correct values should be.

To use a disk to hold files, the operating system still needs to record its own data structures on the disk. The first step is to partition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk. After partitioning, the second step is logical formatting (creation of a file system). The OS stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space and an initial empty directory. To increase efficiency, most file systems group blocks together into larger chunks, frequently called clusters. Disk I/O is done via blocks, but file system I/O is done via clusters.

- ### BOOT BLOCK:

For a computer to start running-for instance, when it is powered up or rebooted-it must have an initial program or bootstrap program to run. It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system. The bootstrap program finds the operating system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution. The bootstrap is stored in ROM. ROM is read only so it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips. Most systems store a tiny bootstrap loader program in the boot ROM whose only job is to bring in a full bootstrap program from disk. The full bootstrap program can be changed easily: A new version is simply written onto the disk. The full bootstrap program is stored in the boot blocks at a fixed location on the disk. A disk that has a boot partition is called a boot disk or system disk. The code in the boot ROM instructs the disk controller to read the boot blocks into memory and then starts executing that code. The full bootstrap

program is more sophisticated than the bootstrap loader in the boot ROM; it is able to load the entire OS from a non-fixed location on disk and to start the operating system running.

- **BAD BLOCKS:**

Because disks have moving parts and small tolerances, they are prone to failure. Sometimes the failure is complete; in this case, the disk needs to be replaced its contents restored from backup media to the new disk. More frequently, one or more sectors become defective. Most disks even come from the factory with bad blocks. Depending on the disk and controller in use, these blocks are handled in a variety of ways.

On simple disks, such as some disks with IDE controllers, bad blocks are handled manually. If format finds a bad block it writes as a special value in corresponding FAT entry to tell the allocation routine not to use that block. If blocks go bad during normal operation then a special program (chkdsk) must be run manually to search for the bad blocks and to lock them away as before. Data that resided on the bad blocks usually are lost.

The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level formatting at the factory and is updated over the life of the disk. Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding. As an alternative to sector sparing, some controllers can be instructed to replace a bad block by sector slipping.

DISK ALLOCATION

When file is created, storage space is allocated to it. Also, when new data is added in an existing file, file size grows and it needs extra storage space. In a similar way, storage space is released when a file is deleted or data from a file is deleted. An important function of the file system is to manage space on the secondary storage, which includes keeping track of both disk blocks allocated to files and the free blocks available for allocation.

There are two main goals, which should be fulfilled while allocating space on disk to files. These goals are as below:

1. Disk space should be utilized effectively.
2. Files should be accessed quickly.

At the time of space allocation, system must keep track of which disk blocks go with which files. Here the disk is considered as a collection of fixed size of blocks, where size varies from system to system. Most commonly it is of 512 bytes of 1kb in size. These blocks are being numbered starting from 0 or 1 to some maximum.

But, secondary storage introduces two additional problems:

1. slows disk access time and
2. larger number of blocks to deal with

In spite of that, many considerations are similar to both environments, particularly, contiguous and non contiguous allocation of files.

Whenever there is need to allocate storage space to a file one or more disk blocks are allocated to the file. in a similar way, whenever a file is being deleted, allocated blocks will become free for reallocation.

There are three main methods for disk allocation:

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

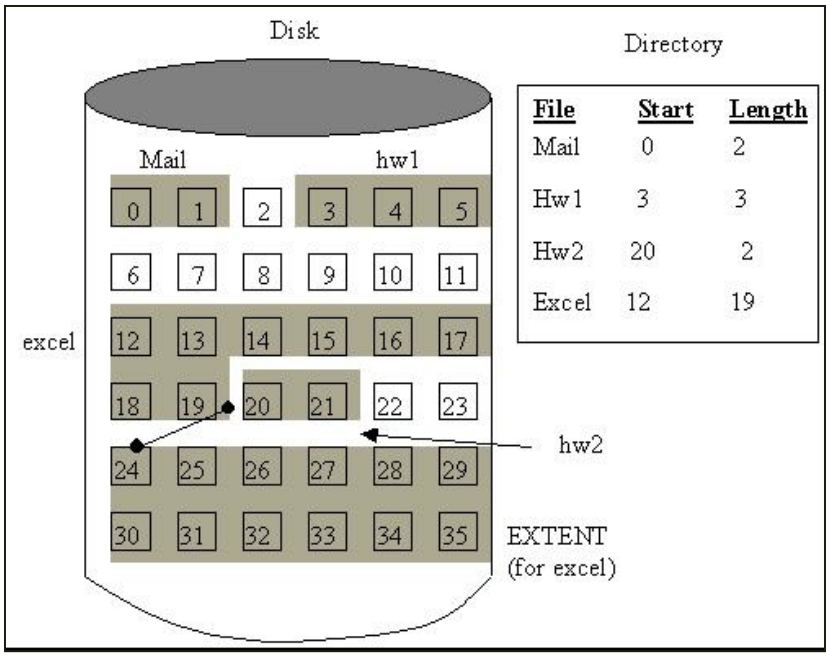
1. CONTIGUOUS ALLOCATION (NOT CONTINUOUS)

In contiguous allocation, files are assigned to contiguous areas of secondary storage. A user specifies in advance the size of the area needed to hold a file to be created. If the desired amount of contiguous space is not available, the file cannot be created.

Each file occupies asset of contiguous blocks on the disk. Thus, on a disk block of 1kb, a file of 50 kb would contain 50 consecutive disk blocks. Whereas, if block size is 2kb; it will occupy 25 consecutive blocks. When a file is created, a disk is searched to find out a chunk of free memory having enough size to store a file. If such chunk is found, required memory is allocated. The directory entry contains file name, starting block number and length of a file.

File Name	Starting Block #	Length

This method is widely used on CD-ROMs. Here, all the files sizes are known in advance. Also, they will never change during subsequent uses of CD-ROM. Figure shown besides is depicting such allocation for 4 different file.



Advantages:

1. Simple to implement. Information here required is only two things; one starting block #; and second, length of file as a total numbers of blocks.
2. All successive records of a file are normally physically adjacent to each other. This increases the accessing speed of records. It means that if records are scattered through the disk it's accessing will be slower. Accessing a file which has been contiguously allocated is fairly easy.

Disadvantages:

1. Finding free space for a new file is time consuming. This requires searching an entire disk until required free memory is found.
2. If size of an existing file increases, it may not be possible to accommodate such extension.
3. External fragmentation is possible. When file is deleted, its blocks are free leaving hole on the disk. With time, disk will consist of files and holes. Such hole may be too small to accommodate new files and will waste space on the disk. It is known as **external fragmentation**. Solution for this problem is defragmentation or compaction the file.

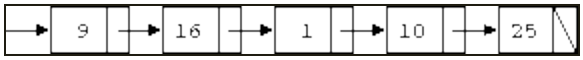
2. LINKED ALLOCATION

Each file is a linked list of disk blocks. Each linked block contains pointer to the next block in the list. These disk blocks may be scattered anywhere on the disk. Directory entry contains the start and last block numbers in a linked list. The directory entry structure is shown below:

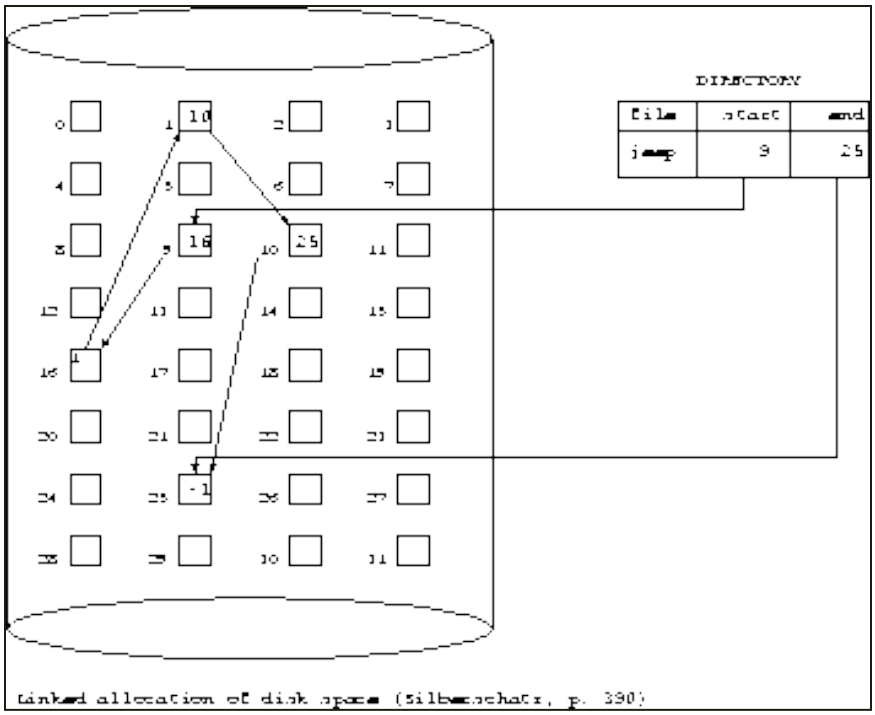
File Name	Starting Block#	Last Block#

When a new file is created; a new directory entry is created. Initially it contains 'null' as both the block number. A write to the file causes free data blocks to be added to the file; such blocks are added t the end of linked list. Directory entry is updated on each such occasion. To read a file, all blocks are read by following the pointers from block to block. Following figure is depicting the linked allocation.

In the figure beside; to reach block # 10, it is required to traverse through block # 9, 16 and 1.



An important variation on the linked allocation method, called File Allocation Table (FAT), is used by the MS-DOS and older versions of Windows Operating Systems.



Linked allocation of disk space (Silberschatz, p. 330)

Advantages:

1. It does not suffer from external fragmentation.
2. Any free disk block can be allocated to a file. Such block does not need to be a consecutive block as in previous method. So, disk space can be utilized effectively.

Disadvantages:

1. File access is time consuming. It is required to access all the data blocks in a linked list to reach some particular block.
2. Random access is not possible directly.
3. Extra space is required for pointers in each data block.

3. LINKED ALLOCATION

In linked allocation, pointers to various disk blocks are scattered on disk among various disk blocks. |Due to this reason, linked allocation cannot support efficient direct access. Indexed allocation solves this problem.

It brings all the pointers together into one location; the Index Block. Each file contains its own index block. An index block is an array of 'disk block addresses'. The 'ith' entry in the index block points to the 'ith' block of the file. Directory entry contains file name and the index block #. This looks as shown below:

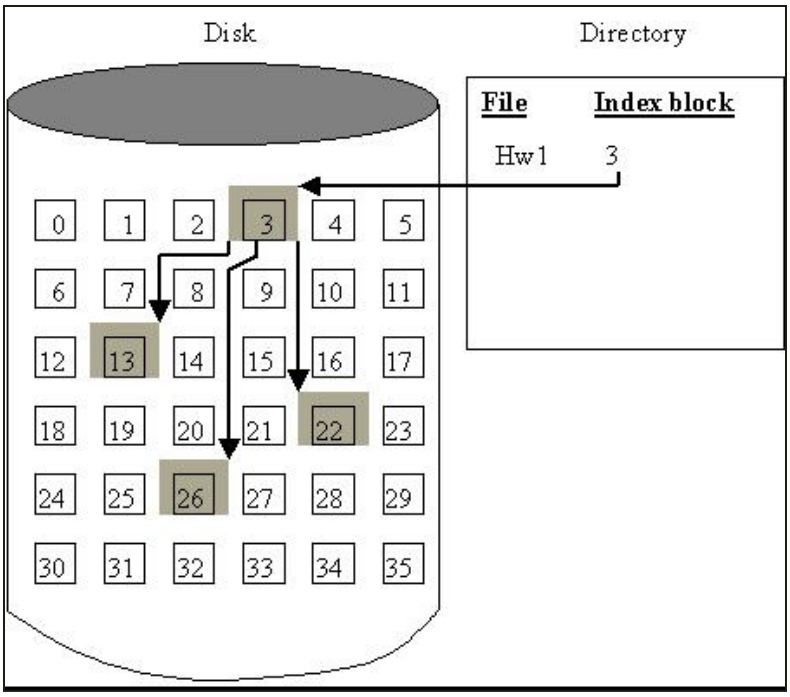
File Name	Index Block#

When new file is created, a new directory entry is created. Initially all pointers in index block are set null. When the 'ith' block is first written, a free disk block is allocated and its address is put in the 'ith' entry in the index block.

The following figure depicts the indexed allocation for the file 'Hw1'. The directory entry for the index block is done as block #3 for this file. This index block entry is composed of all other block # entries which are having the data contents of the file Hw1.

The index entry consists of all the block entries which consists of the data contents.

13	26	22



Advantages:

1. It does not suffer from external fragmentation.
2. Direct access is efficient.

Disadvantages:

1. It suffers from wasted space. Index block may be partially filled. This wastes remaining memory space of an index block. For example, if file contains two data blocks, then index block will have only two entries to point to theses two data blocks, remaining entire index block will be wasted,
2. Maximum allowable file size depends on the size of an index block.

Solutions:

In this allocation method the main problem is the size of index block. If it is too large, it may waste space. If it is too small it cannot accommodate enough pointers for a large file. Following are the solutions:

1. Linked Scheme

An index block is normally of one disk block size. For larger files more than one index block can be used by linking them together by a linked list.

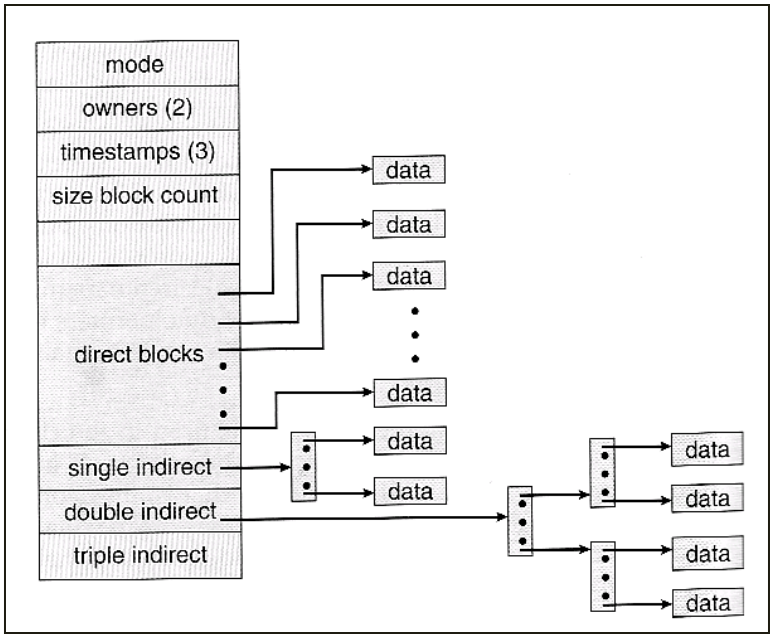
2. Multilevel Index / Hierarchical Indexing

Two or more level of index blocks is used here. First level index blocks point to a set of second level index blocks. Second level index blocks point to disk blocks containing the data. For larger and larger file, levels can be increased.

3. Combined Scheme

In this solution, partial entries from indirect block point to direct block containing file data. Partial entries point to indirect blocks, which are index blocks. They do not contain file data, but they contain address of disk blocks (pointers) that do contain file data. UNIX Os is using such type of combined scheme.

Figure shown besides is an example of such combined scheme which is implemented in the UNIX operating system for storing the inode.



DISK SCHEDULING

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth. The access time has two major components. The **SEEK TIME** is the time for the disk arm to move the heads to the cylinder containing the desired sector. The **LATENCY** is the additional time for the disk to rotate the desired sector to the disk head. The disk **BANDWIDTH** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer. We can improve both the access time and the bandwidth by managing the order in which disk I/O requests are serviced. Whenever a process needs I/0 to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information:

- Whether this operation is input or output
- What the disk address for the transfer is
- What the memory address for the transfer is
- What the number of sectors to be transferred is

If the desired disk drive and controller are available, the request can be serviced immediately. If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive. For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system chooses which pending request to service next. How does the operating system make this choice? Any one of several disk-scheduling algorithms can be used, and we discuss them next.

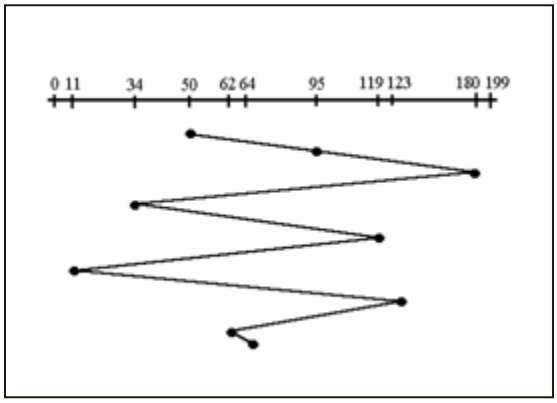
TYPES OF DISK SCHEDULING ALGORITHMS

Although there are other algorithms that reduce the seek time of all requests, I will only concentrate on the following disk scheduling algorithms:

1. First Come-First Serve (FCFS) 2. Shortest Seek Time First (SSTF) 3. Elevator (SCAN)
4. Circular SCAN (C-SCAN) 5. LOOK 6. C-LOOK

These algorithms are not hard to understand, but they can confuse someone because they are so similar. What we are striving for by using these algorithms is keeping Head Movements (# tracks) to the least amount as possible. The less the head has to move the faster the seek time will be. I will show you and explain to you why C-LOOK is the best algorithm to use in trying to establish less seek time.
Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199 let us now discuss the different algorithms.

1. First Come First Serve (FCFS)

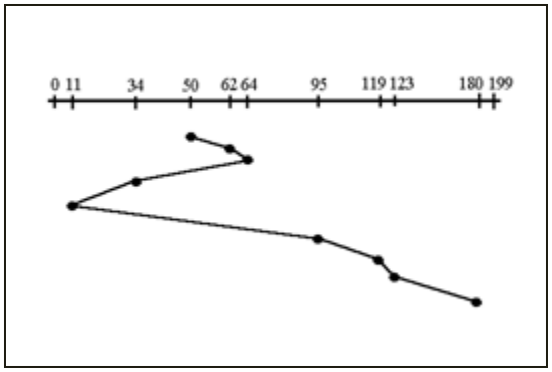


All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next.

For this case it went from 50 to 95 to 180 and so on. From 50 to 95 it moved 45 tracks. If you tally up the total number of tracks you will find how many tracks it had to go through before finishing the entire request. In this example, it had a total head movement of 640 tracks. The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and then back to track 11 to 123 then to 64. As you will soon see, this is the worse algorithm that one can use.

Advantage Simple and Fair to all requests
Disadvantage Not efficient, because the average seek time is very high. It suffers from zigzag effect.

2. Shortest Seek Time First (SSTF)



Selects the request with the minimum seek time from the current head position. Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances. It's biased in favor of the middle cylinders requests. SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

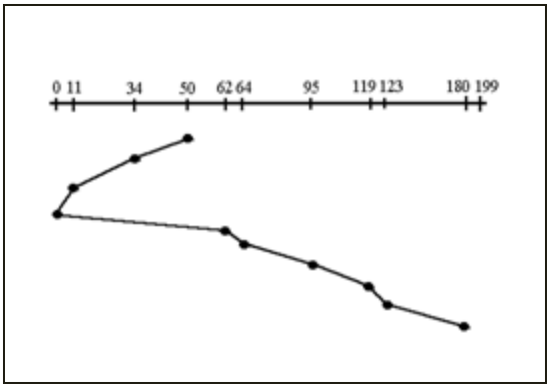
In this case request is serviced according to next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the processes are taken care of. For example the next case would be to move from 62 to 64 instead of 34 since there are only 2 tracks between them and not 18 if it were to go the other way. Although this seems to be a better service being that it moved a total of 236 tracks, this is not an optimal one. There is a great chance that starvation would take place. The reason for this is if there were a lot of requests close to each other the other requests will never be handled since the distance will always be greater.

Advantage More efficient than FCFS
Disadvantage Starvation is possible for requests involving longer seek time

Elevator Algorithms

Theses algorithms are based on the common elevator principle. Four combinations of Elevator algorithms: **SCAN**, **LOOK**, **C-SCAN** and **C-LOOK**. They services in both directions or in only one direction. They operates until last cylinder or until last I/O request encountered.

3. Elevator (SCAN)



This approach works like an elevator does. The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues. It moves in both directions until both ends. IT tends to stay more at the ends so more fair to the extreme cylinder requests. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up. This process moved a total of 230 tracks. Once again this is more optimal than the previous algorithm, but it is not the best.

Advantage

More efficient than FCFS

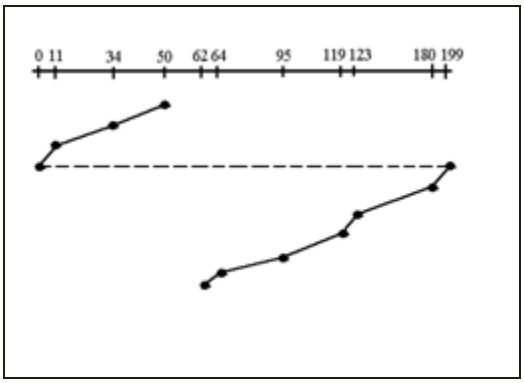
Also there is no starvation for any requests

Disadvantage

Require extra head movement between two extreme points. For example, after servicing 5^{th} cylinder there is no need to visit the 0^{th} cylinder. Though, this algorithm visits the end points.

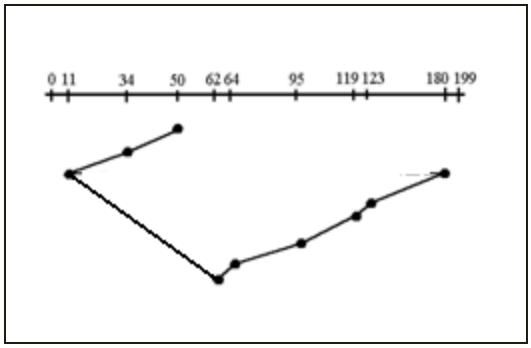
Not so fair; cylinders which are just behind head will wait longer

4. Circular Scan (C-SCAN)



Circular scanning works just like the elevator to some extent. The head moves from one end of the disk to the other, servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip. It treats the cylinders as a circular list that wraps around from the last cylinder to the first one. It provides a more uniform wait time than SCAN; it treats all cylinders in the same manner. It begins its scan toward the nearest end and works its way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction. Keep in mind that the huge jump doesn't count as a head movement. The total head movement for this algorithm is only 187 tracks, but still this isn't the more sufficient.

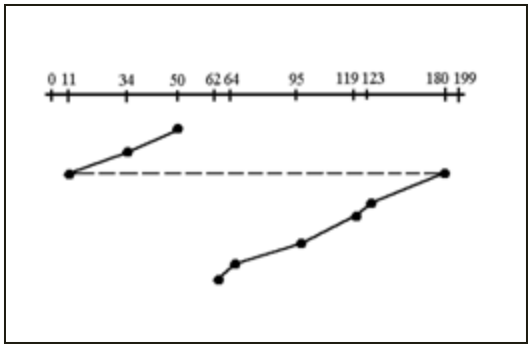
5. LOOK



The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues. It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.

In this example head will move from 11 to 62 rather going to 0 from 11.

6. Circular LOOK(C-Look)



It is an enhanced Look version of C-Scan. Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk. Scan versions have a larger total seek time than the corresponding Look versions.

This is just an enhanced version of C-SCAN. In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request. C-SCAN had a total movement of 187 but this scan (C-LOOK) reduced it down to 157 tracks.

From this you were able to see a scan change from 644 total head movements to just 157. You should now have an understanding as to why your operating system truly relies on the type of algorithm it needs when it is dealing with multiple processes.

Selecting Disk Scheduling Algorithms

Here some factors are given which effect the performance of a disk scheduling algorithm as shown below:

1. **Number and types of Requests:**
   - Performance of a disk scheduling algorithm heavily depends upon the number and type of request
   - For example, if the device queue contains only one choice is available, all algorithms will behave like FCFS.
2. **File Allocation Method:**
   - File allocation method also has good contribution in deciding performance.
   - For example, when a contiguously allocated file is accessed, the disk requests are normally close to each other. Whereas for a linked or indexed file, disk blocks are scattered over disk, and disk head requires great movements.
3. **Location of directories and index blocks:**
   - Every file must be opened before use. For this, there is a need to search directory structure to determine location of files. So, directories are accessed frequently.
   - If the directory entry is on the first cylinder, and file's data are on the final cylinder, then disk head required to move the entire width of the disk.

These are the main three factors that must be observed at the time of selecting the algorithm. An average seek time with FCFS is very high. SCAN required unnecessary head movements between two end points. So, in default case, either SSTF or LOOK is reasonable choice as a disk scheduling algorithm.