An Operating System provides services to both the users and to the programs.

It provides programs an environment to execute.
It provides users the services to execute the programs in a convenient manner.
Following are a few common services provided by an operating system −

Program execution
I/O operations
File System manipulation
Communication
Error Detection
Resource Allocation
Protection
Program execution
Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −

Loads a program into memory.
Executes the program.
Handles program's execution.
Provides a mechanism for process synchronization.
Provides a mechanism for process communication.
Provides a mechanism for deadlock handling.
I/O Operation
An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

I/O operation means read or write operation with any file or any specific I/O device.
Operating system provides the access to the required I/O device when required.
File system manipulation
A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

Program needs to read a file or write a file.
The operating system gives the permission to the program for operation on file.
Permission varies from read-only, read-write, denied and so on.
Operating System provides an interface to the user to create/delete files.
Operating System provides an interface to the user to create/delete directories.
Operating System provides an interface to create the backup of file system.
Communication
In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

Two processes often require data to be transferred between them
Both the processes can be on one computer or on different computers, but are connected through a computer network.
Communication may be implemented by two methods, either by Shared Memory or by Message Passing.
Error handling
Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

The OS constantly checks for possible errors.
The OS takes an appropriate action to ensure correct and consistent computing.
Resource Management
In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

The OS manages all kinds of resources using schedulers.
CPU scheduling algorithms are used for better utilization of CPU.
Protection
Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

The OS ensures that all access to system resources is controlled.

The OS ensures that external I/O devices are protected from invalid access attempts.
The OS provides authentication features for each user by means of passwords.
Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System
Linux Operating System has primarily three components

Kernel − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

System Library − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.

System Utility − System Utility programs are responsible to do specialized, individual level tasks.

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

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

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

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

Open Source − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

Multi-User − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

Multiprogramming − Linux is a multiprogramming system means multiple applications can run at same time.

Hierarchical File System − Linux provides a standard file structure in which system files/ user files are arranged.

Shell − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.

Security − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture
The following illustration shows the architecture of a Linux system −

Linux Operating System Architecture
The architecture of a Linux System consists of the following layers −

Hardware layer − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

Kernel − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

Shell − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.

Utilities − Utility programs that provide the user most of the functionalities of an operating systems.
What is Thread?
A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a lightweight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.

Single vs Multithreaded Process

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

Threads minimize the context switching time.
Use of threads provides concurrency within a process.
Efficient communication.
It is more economical to create and context switch threads.
Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

Threads are implemented in following two ways −

User Level Threads − User managed threads.

Kernel Level Threads − Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

User level thread

Advantages

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

Disadvantages

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

Kernel Level Threads

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

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

Advantages
Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
Kernel routines themselves can be multithreaded.
Disadvantages
Kernel threads are generally slower to create and manage than the user threads.
Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.
Multithreading Models
Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

Many to many relationship.
Many to one relationship.
One to one relationship.
Many to Many Model
The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

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

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

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