Lecture1

Introduction to Operating Systems

Introduction to Operating System

An operating system acts as an **intermediary** between the user of a computer and computer hardware. The **purpose** of an operating system is to provide an environment in which a **user can execute programs** in a **convenient** and **efficient** manner.

An operating system is a **software** that **manages** the **computer hardware**. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to **prevent** user programs from **interfering** with the proper operation of the system.

Generations of Operating System

The First Generation (1940 to early 1950s)

When the first electronic computer was developed in **1940**, **it was created without any operating system**. In early times, **users have full access to the computer machine** and **write a program for each task in absolute machine language**. The programmer can perform and solve only simple mathematical calculations during the computer generation, and this calculation does not require an operating system.

The Second Generation (1955 - 1965)

The first operating system (OS) was created in the early **1950s**. The second-generation operating system was **based on a single stream batch processing system** because it **collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine**.

The Third Generation (1965 - 1980)

During the late **1960s**, operating system designers were very capable of developing a new operating system that could simultaneously perform **multiple tasks** in a **single computer** program called **multiprogramming**.

The Fourth Generation (1980 - Present Day)

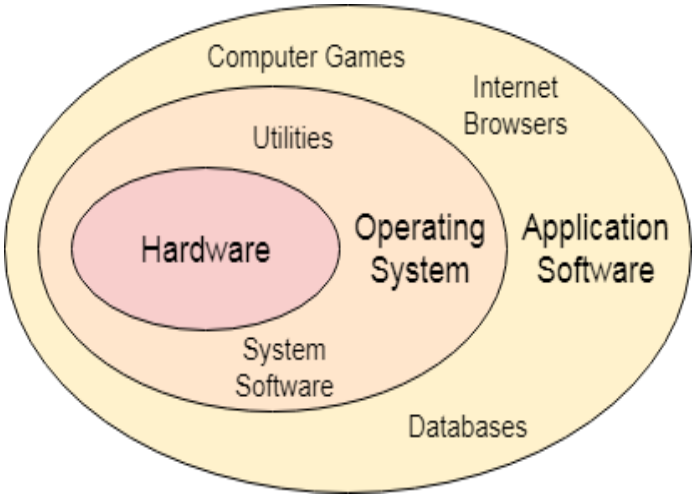
The fourth generation of operating systems is related to the development of the **personal computer**. **However**, Microsoft created the first **window operating system** in **1975**. **Therefore**, they introduced the **MS-DOS** in **1981**.

Besides the **Windows** operating system, **Apple** is another popular operating system built in the **1980s**.They named the operating system **Macintosh OS or Mac OS**.

Operating System Definition and Function

In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.

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



An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the **execution** of all the **processes**, **Resource Allocation**, **CPU management**, **File Management** and many other tasks.

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

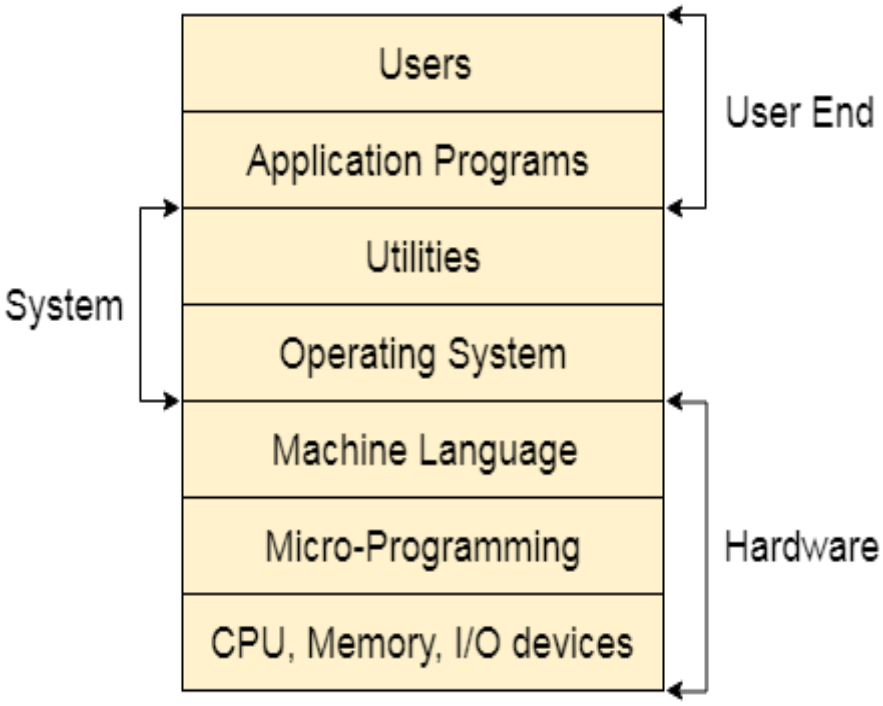
Functions of Operating system

Operating system performs three functions:

1. **Convenience:** An OS makes a computer more convenient to use.
2. **Efficiency:** An OS allows the computer system **resources** to be used in an **efficient** manner.
3. **Ability to Evolve:** An OS should be constructed in such a way as to permit the effective development, testing and introduction of **new system functions** at the same time **without interfering with service**.

Structure of a Computer System

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



What does an Operating system do?

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

I/O System Management

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

The I/O system of an OS works by taking **I/O request** from an **application software** and **sending** it to the **physical device**, which could be an input or output device then it takes whatever **response** comes back from the device and sends it to the **application**.

Components of I/O Hardware

- I/O Device
- Device Driver
- Device Controller

I/O Device: I/O devices such as storage, communications, user-interface, and others communicate with the computer via signals sent over wires or through the air. Devices connect with the computer via ports, e.g. a serial or parallel port. A common set of wires connecting multiple devices is termed a bus.

I/O devices can be divided into two categories:

- **Block devices** – the device driver communicates by **sending** entire **blocks of data**. For example, Hard disks, USB cameras, Disk-OnKey etc.
- **Character devices** – the device driver communicates by **sending** and **receiving single characters** (bytes, octets). For example, serial ports, parallel ports, sounds card etc.

Device Driver: Device drivers are **software modules** that can be plugged into an OS to handle a particular device.

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

Processor

It is a **programmable device** that **takes input**, perform some **arithmetic and logical operations** and **produce some output**.

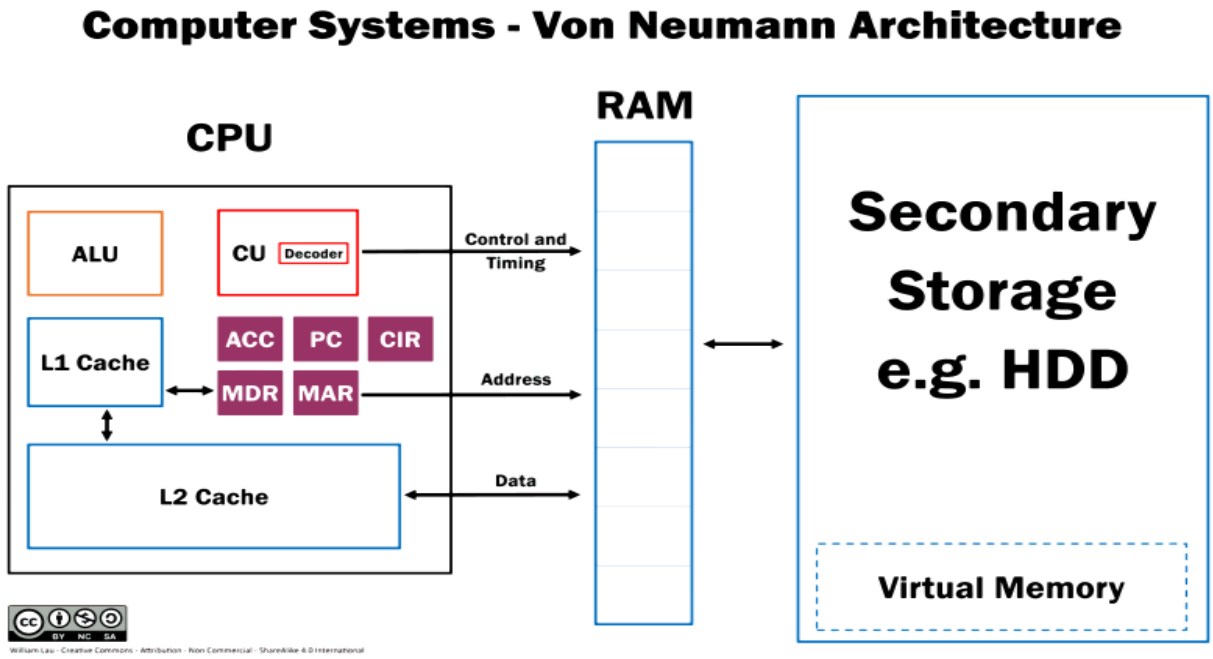
Basics of a Processor

A processor takes a **bunch of instructions** in machine language and executes them, telling the processor what it has to do.

Processors performs **three** **basic operations** while executing the instruction:

1. It performs some basic operations like **addition**, **subtraction**, **multiplication**, **division** and some logical operations using its Arithmetic and Logical Unit (**ALU**).
2. Data in the processor can **move** from one **location to another**.
3. It has a **Program Counter (PC) register** that stores the address of next instruction based on the value of PC.

A typical processor structure looks like this.



Basic Processor Terminology

- **Control Unit (CU)**
  A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches the code for instructions and controlling how data moves around the system.
- **Arithmetic and Logic Unit (ALU)**
  The arithmetic logic unit handles all the calculations the CPU may need.
- **Main Memory Unit (Registers)**
  1. **Accumulator (ACC):** Stores the results of calculations made by **ALU**.
  2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The **PC** then **passes** this next address to Memory Address Register (**MAR**).
  3. **Memory Address Register (MAR):** It stores the **memory locations** of instructions that need to be **fetched** from memory or **stored** into memory.
  4. **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
  5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
  6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register **IBR**.
- **Input/Output Devices –** Program or data is read into main memory from the ***input device*** or secondary storage under the control of CPU input instruction. ***Output devices*** are used to output the information from a computer.
- **Buses –** Data is transmitted from one part of a computer to another, **connecting** all major internal **components** to the **CPU and memory**, by the means of Buses. Types:
  1. **Data Bus (Data):** It carries data among the **memory** unit, the **I/O devices**, and the **processor**.
  2. **Address Bus (Address)**: It carries the address of data (not the actual data) between **memory** and **processor**.
  3. **Control Bus (Control and Timing):** It carries **control commands** from the **CPU** (and status signals from other devices) in order to control and coordinate all the activities within the computer.

I/O Modules

The **method** that is used to **transfer** information **between main memory and external I/O devices** is known as the I/O interface, or I/O modules.

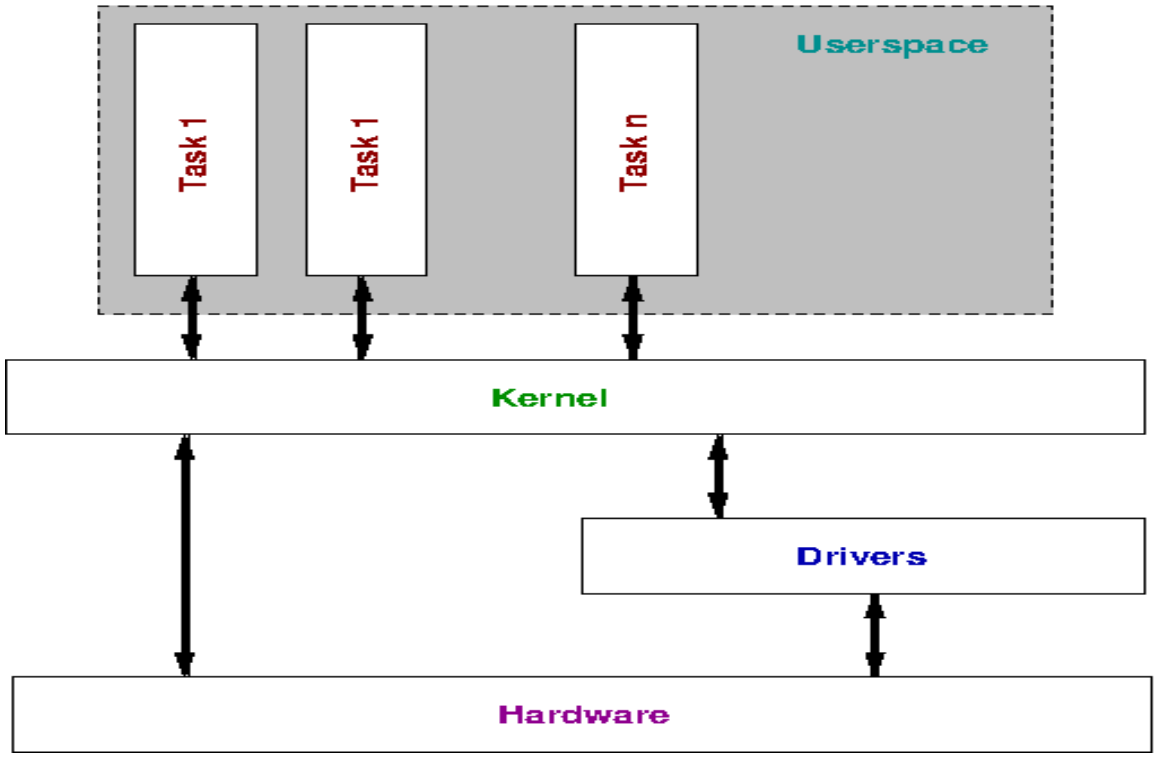
Mode of Transfer:

Data transfer to and from the peripherals may be done in any of the three possible ways

1. **Programmed I/O:** is the result of the I/O instructions written in the program's code.
2. **Interrupt- initiated I/O:** using an interrupt facility and special commands to issue an interrupt request signal whenever data is available from any device.
3. **Direct memory access(DMA)**: The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

The Kernel

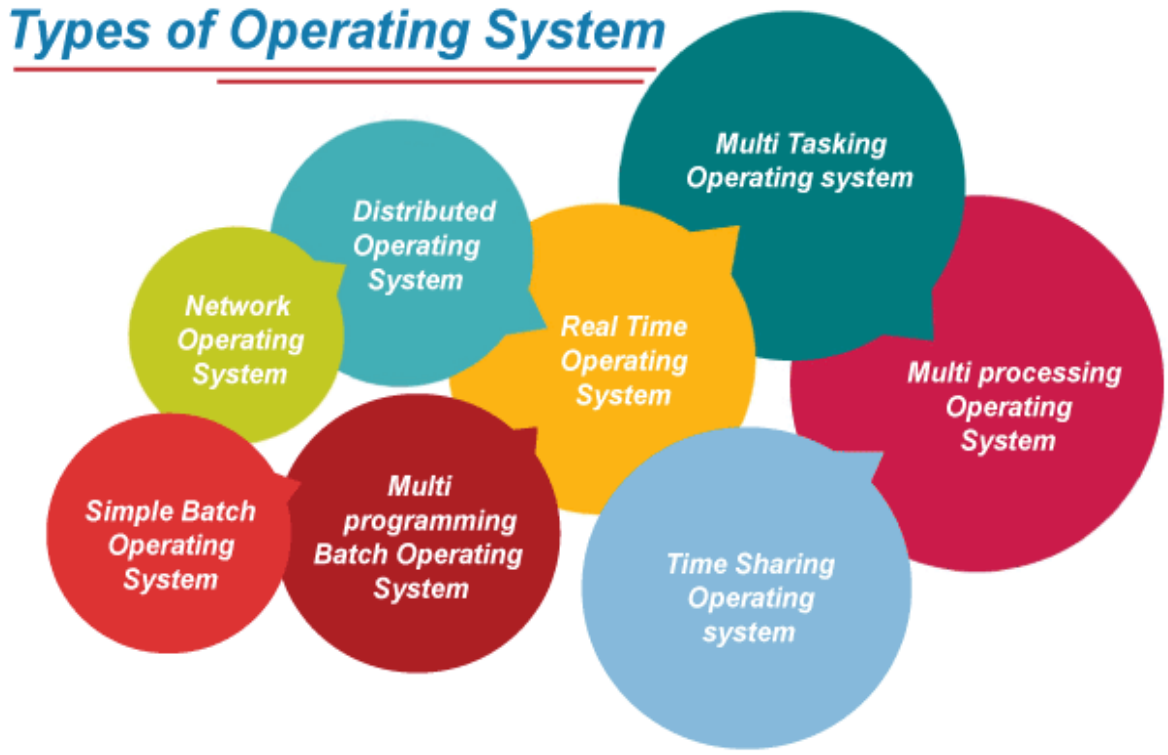
A kernel is the core component of an operating system. Using **interprocess communication** and **system calls**, it acts as a **bridge** between **applications** and the **data processing** performed at the **hardware level**.



system call (syscall)

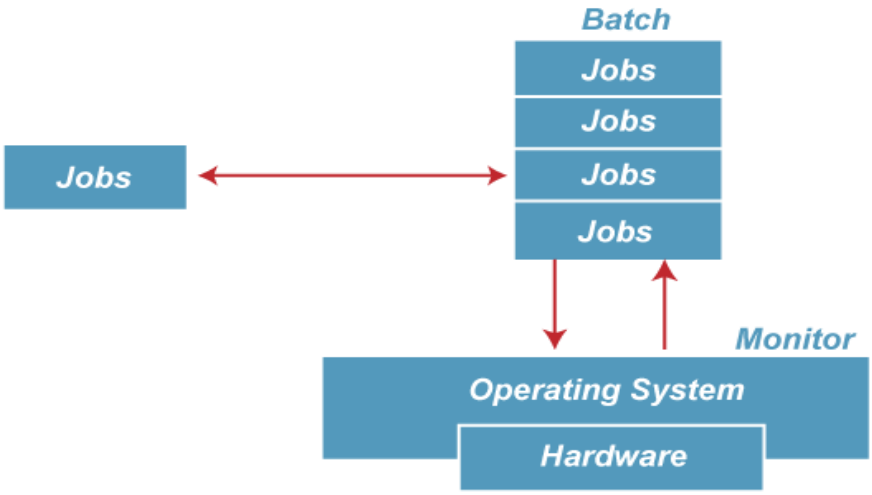
is the programmatic way in which a computer program **requests a service** from the **operating system** on which it is executed.

Types of Operating System



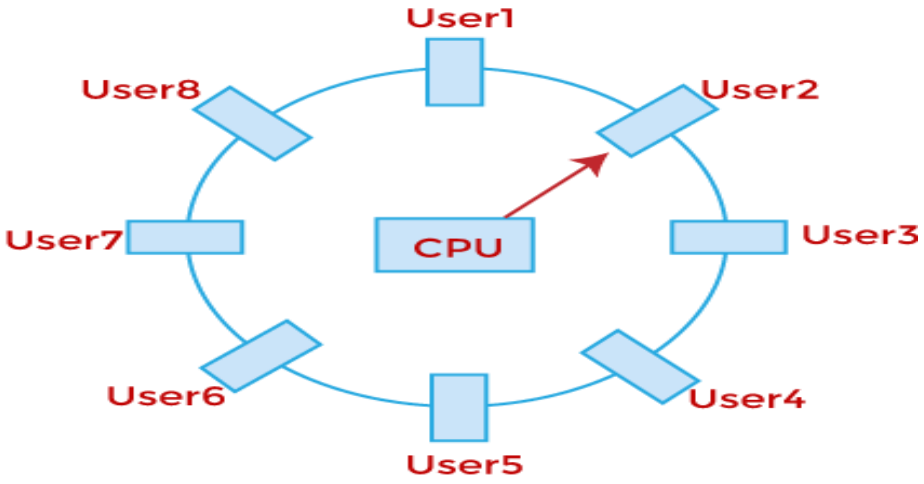
1. **Batch Operating System**

   The system put **all of the jobs in a queue on the basis of first come first serve** and then executes the jobs **one by one**.



2. Time-Sharing Operating System

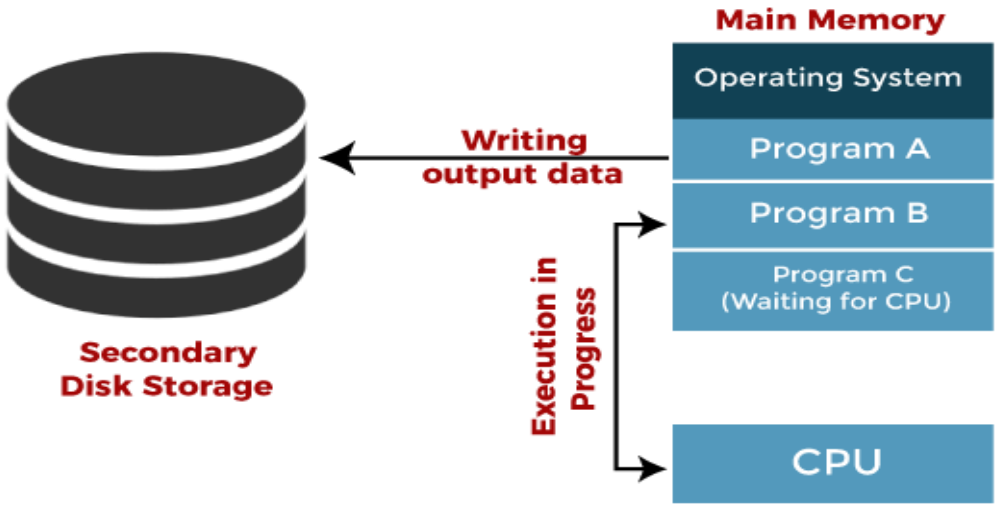
A time-sharing operating system **allows many users to be served simultaneously**, so sophisticated CPU scheduling schemes and Input/output management are required.



Timesharing in case of 8 users

3. Multiprogramming Operating System

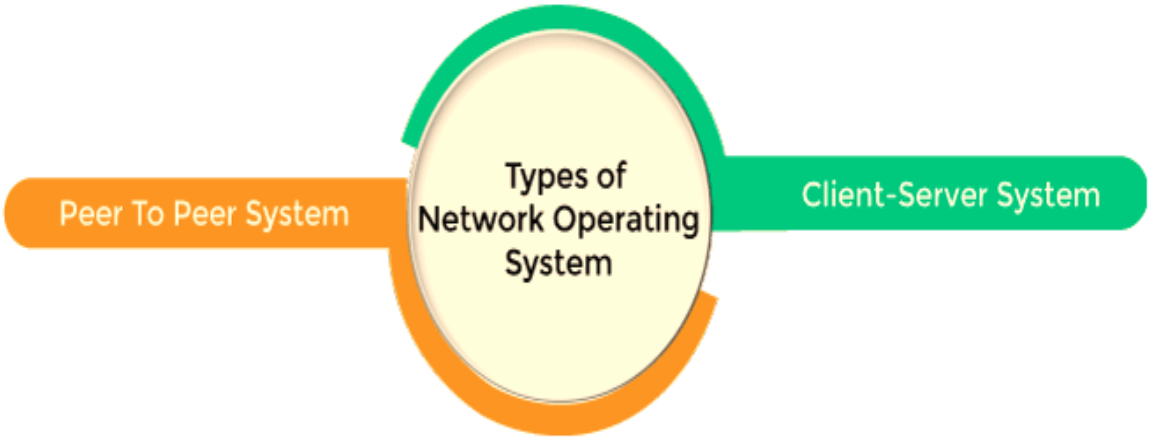
Multiprogramming is an extension to **batch processing where the CPU is always kept busy**. Each process needs two types of system time: **CPU time** and **IO time.**



Jobs in multiprogramming system

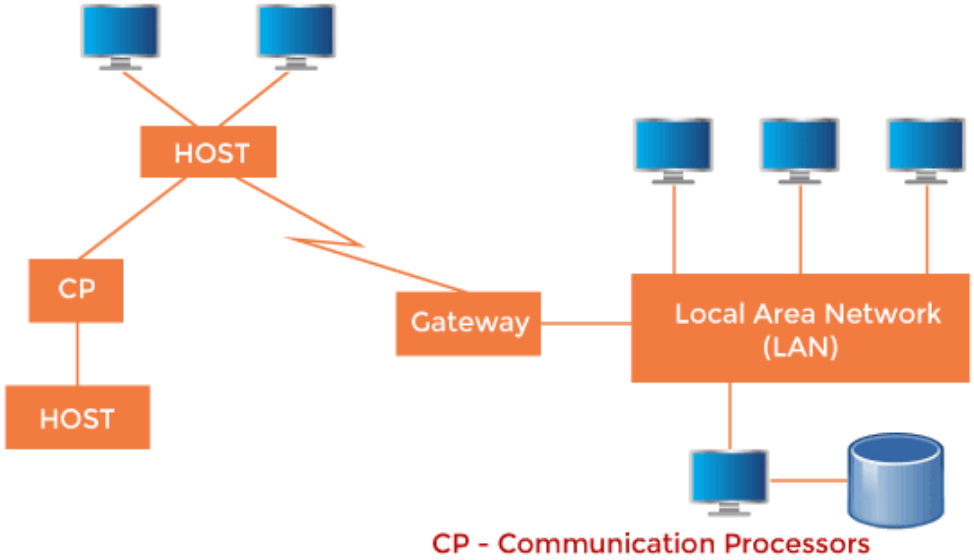
4. Network Operating System

An Operating system, which **includes software and associated protocols** to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.



5. Distributed Operating System

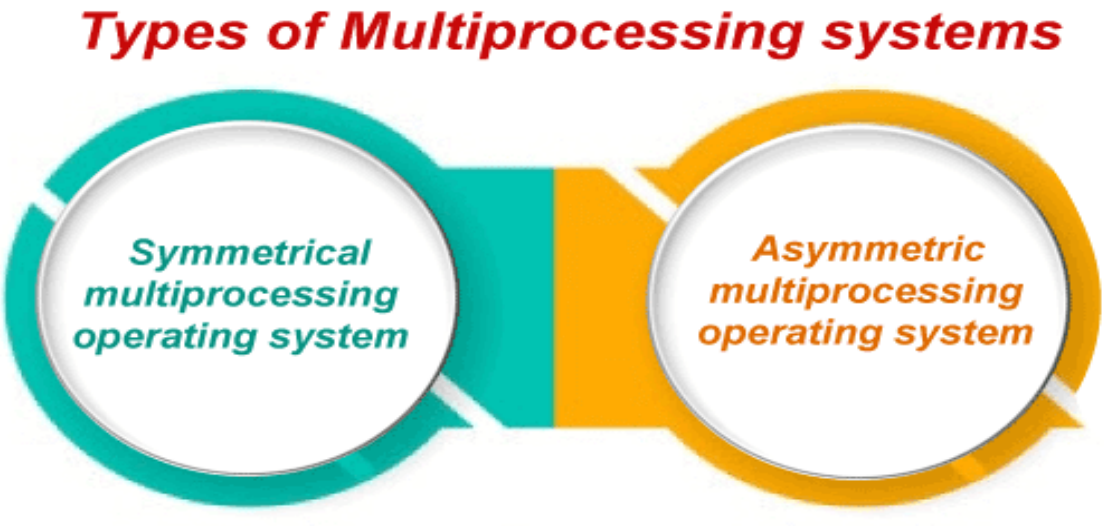
The Distributed Operating system is not installed on a single machine, it is **divided into parts, and these parts are loaded on different machines**. A part of the distributed Operating system is installed on each machine to make their communication possible.



A Typical View of a Distributed System

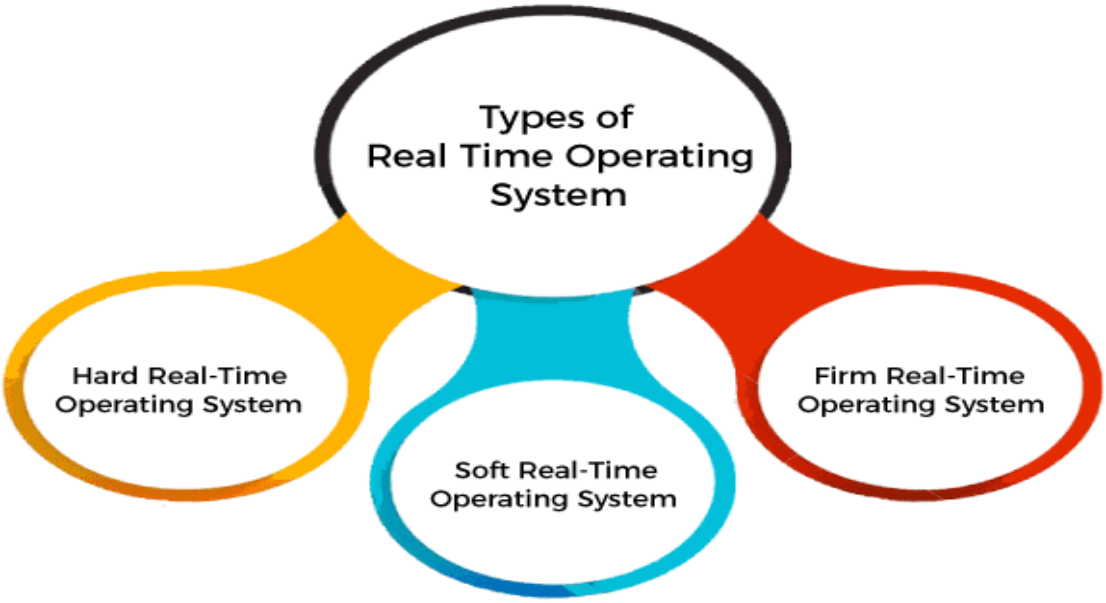
6. **Multiprocessing Operating System**

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



7. **Real-Time Operating System**

In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed.



Mobile Operating System

A **mobile operating system** is an operating system that helps to **run other application software on mobile devices**. It is the same kind of software as the computer operating systems like **Linux** and **Windows**, but now they are **light** and **simple** to some extent.

The operating systems found on smartphones include:

- Symbian OS
- iPhone OS
- RIM's BlackBerry
- Windows Mobile
- Palm WebOS
- Android

Operating System	Kernel
Symbian OS	Symbian OS kernel
RIM's BlackBerry	QNX Neutrino
Windows Mobile	Windows NT kernel
Palm WebOS	Linux kernel
Android	Linux kernel
iOS	XNU kernel.

Popular platforms of the Mobile OS

1. **Android OS:** The Android operating system is the most popular operating system today. It is a mobile OS based on the **Linux Kernel** and **open-source software**. The android operating system was developed by **Google**.
2. **Bada (Samsung Electronics):** Bada is a **Samsung** mobile operating system that was launched in 2010. offers many mobile features, such as **3-D graphics**, application installation, and multipoint-touch.(NOW merge with the open-source operating system **Tizen**)
3. **BlackBerry OS:** is a mobile operating system developed by **Research In Motion (RIM)**. This operating system was designed specifically for **BlackBerry handheld devices**.
4. **iPhone OS / iOS:** The iOS was developed by the **Apple inc** for the use on its device. It is a very **secure** operating system.
5. **Symbian OS:** The Symbian operating system is based on the **java language**. It combines **middleware** of wireless communications and personal information management (**PIM**) functionality. **Nokia** was the first company to release **Symbian OS** on its mobile phone.
6. **Windows Mobile OS:** The window mobile OS is a mobile operating system that was developed by **Microsoft**. It was designed for the **pocket PCs** and **smart mobiles**.
7. **Harmony OS:** The harmony operating system is the latest mobile operating system that was developed by **Huawei** for the use of its devices. It is designed primarily for **IoT** devices.
8. **Palm OS:** The palm operating system is a mobile operating system that was developed by **Palm Ltd** for use on personal digital assistants (**PADs**). It was introduced in 1996. **Palm OS** is also known as the **Garnet OS**.
9. **WebOS (Palm/HP):** The WebOS is a mobile operating system that was developed by Palm. It based on the Linux Kernel. The **HP** uses this operating system in its mobile and touchpads.

Android Application fundamentals

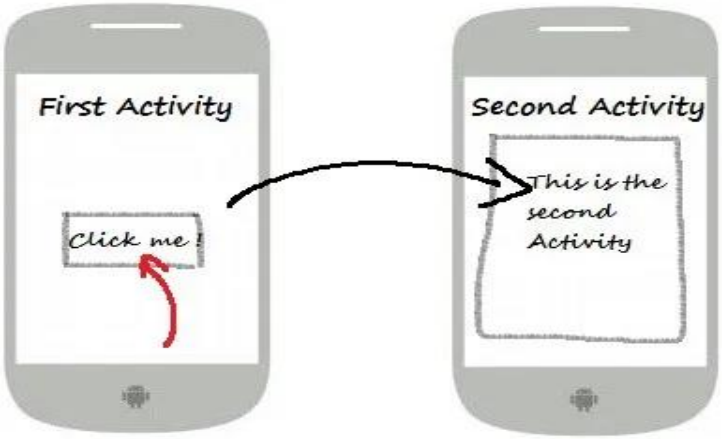
App components

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four types of app components:

1. **Activities**

   An *activity* is the entry point for interacting with the user. It represents a single screen with a user interface.



2. **Services**

   A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface.

   There are two types of services that tell the system how to manage an app:

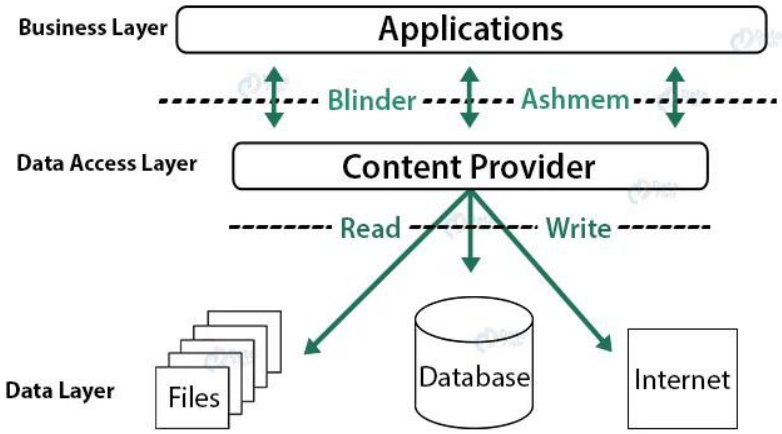
   - **Started services**: tell the system to keep them running until their work is completed.
   - **Bound services**: A bound service provides an API to another process, and the system knows there is a dependency between these processes.

3. **Broadcast receivers**

Broadcast Receiver is a component that responds to broadcast messages from another application or the same system.

4. **Content providers**

Content Provider is a component that allows applications to share data among multiple applications.



Working of Android Content Provider

5. **Intents**

It is an inter-application message passing framework for communication between android components.

Android Activity Lifecycle

An activity can have **four states**, which are:

1. **Running State**: An activity is in the running state if it's shown in the foreground of the users' screen.
2. **Paused State:** When an activity is not in the focus but is still alive for the user, it's in a paused state. The activity comes in this state when some other activity comes in with a higher position in the window.
3. **Resumed State**: It is when an activity goes from the paused state to the foreground that is an active state.
4. **Stopped State:** When an activity is no longer in the activity stack and not visible to the users.

Android Activity Methods

In Android, we have the following **7 callback methods** that activity uses to go through the four states:

1. **onCreate()**
   The Android **oncreate()** method is called at the very start when an activity is created.
2. **onStart()**
   The Android **onstart**() method is invoked as soon as the activity becomes visible to the users
3. **onPause()**
   The Android **onPause**() method is invoked when the activity doesn't receive any user input and goes on hold.
4. **onRestart()**
   The Android **onRestart**() method is invoked when activity is about to start from the stop state.
5. **onResume()**
   The Android **onResume**() method is invoked when the user starts interacting with the user. This callback method is followed by **onPause()**.
6. **onStop()**

The Android **onStop**() method is invoked when the activity is no longer visible to the user.

7. onDestroy()

The Android **onDestroy**() is the method that is called when an activity finishes and the user stops using it.