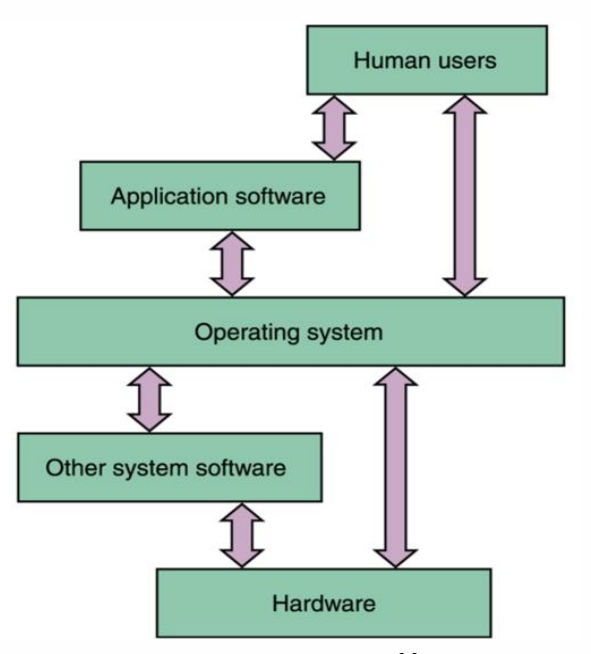
**INTRODUCTION**

**Concepts of OS**

* **Program:** Set of instructions.
* **Software:** Set of programs.

**Types of Software**

* **System software:** Software used in performing **hardware level** tasks.
  + Compiler & interpreter
  + Operating system
  + **Linker:** Links required libraries to the program during execution.
  + **Loader:** Loads required libraries after linker links libraries with program.
* **Utility software:** Set of **pre-built software** in an OS.
  + Calculator
  + Browser
* **Application languages:** Software **made using high-level languages** & has less to do with hardware. **For example:** some games, banking system etc.



**Goals of OS**

* Manage hardware resources efficiently.
* Allow sharing of hardware & software.
* Provide security to user’s programs.
* Provide troubleshoot & error correction etc.

**Generations of OS**

|  |  |  |
| --- | --- | --- |
| **Generation** | **Year** | **Method** |
| **1st** | **1945-1955** | **Vacuum tubes & plug boards** |
| **2nd** | **1955-1965** | **Transistors & batch systems** |
| **3rd** | **1965-1980** | **IC & multi-programming** |
| **4th** | **1980-NOW** | **Personal computers** |

**1st Generation of OS**

Vacuum tubes:-

* Before WW2 **mechanical relay** based computers were used primarily for **calculations**.
* However these were slow.
* After WW2 **first digital computers** were created using **vacuum tubes**.
* These were large & faster but still slow.

Plugboards:-

* It had **no** programming language & **no** OS.
* It was used for simple numerical **calculations**.

Punch cards:-

* Introduced during 50’s.
* Numeric patterns were **punched** **on cards** and fed to the system.
* It improved computing experience.

**2nd Generation of OS**

Transistors:-

* First **commercial** class of computers were made.
* Were also known as **mainframes**.
* Were kept in air-conditioned rooms where user operated them.

Batch systems:-

* Introduced to **reduce time wastage** in operating.
* A tray full of **“jobs”** was collected in input room & read using **magnetic tapes**.

**3rd Generation of OS**

IC:-

* Reaching the 60’s there were **commercial & scientific** computers.
* **IBM** did work of **integrating** these systems.
* These were bit expensive.
* Provided major performance boost over previous generation.

Multi-programming:-

* Processor was **never** idle using this.
* While a job completes its I/O operation, it processes another one.

**4th Generation of OS**

PC:-

* Created with large scale ICs.
* Thousands of transistors/cm2 of **silicon**.
* It let to further development of **network OS & distributed OS**.
* Network OS was earlier used for **copying files** from one machine to another.

**PC Examples**

Mac OS:-

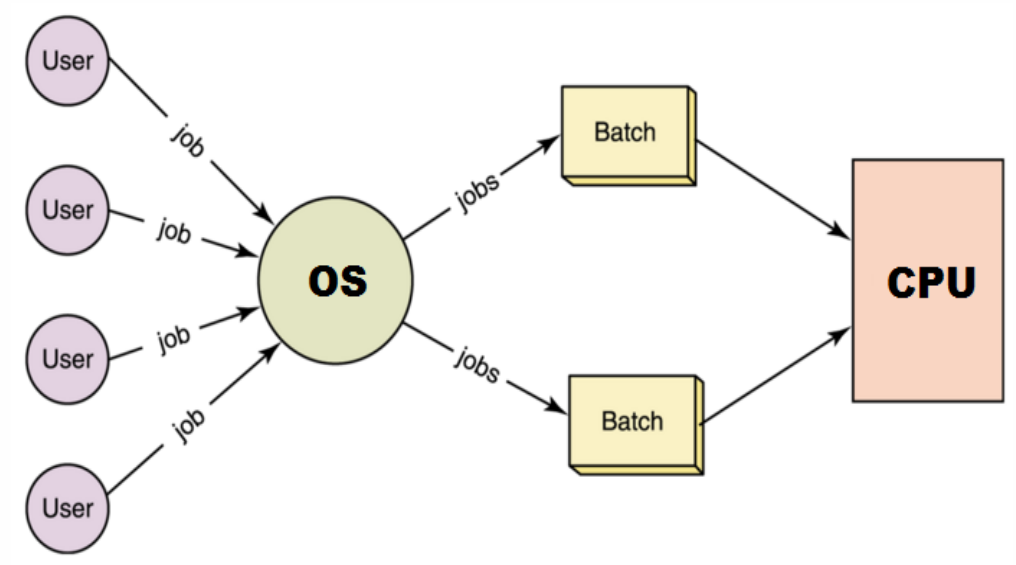
* Apple computers are called **Macintosh**.
* Apple laptops are called **Mac**.
* Recent versions:
  + El Capitan (2015)
  + Yosemite (2014)
  + Mavericks (2013)
  + Mountain Lion (2012)
  + Lion (2011)

**Types of OS**

* Simple Batch OS
* Multiprogramming Batch OS
* Multitasking OS
* Multiprocessor OS
* Distributed OS
* Real-time OS

**Simple Batch OS**

* **No** **direct** interaction b/w user & computer.
* User submits a **job** to the **computer operator** (OS).
* **Job:** Instructions written on card/tape.
* Then the computer operator makes **batches** of these jobs.
* **Batch:** Set of jobs.
* And finally places these batches on an input device.



Disadvantages:-

* No priority-based processing.

**Multiprogramming Batch OS**

* All jobs are submitted to the OS at the **same time**.
* These jobs are loaded into the **main memory**.
* Then one-by-one selected & executed.
* If during execution a job required **peripheral** for further process, then the CPU handles **another job**.

Advantages:-

* Resources (**memory**, **I/O**, **CPU**) are used more efficiently.
* User can submit all jobs **simultaneously**.

Disadvantages:-

* Poor CPU usage as only program is there at a time.
* No user interaction.

**Multitasking OS**

* Also called **Time Sharing System (TTS)**.
* Interacts with user.
* It uses **CPU scheduling** & **multiprogramming**.
* Supports **concurrent access**.
* Each user is given its **time-slice** for executing his job in **Round-Robin Fashion**.
* **Round-Robin:** An algorithm in which each user is given **equal** amount of CPU in sequence.
* Multitasking allows more frequent **context switching**.
* **Context switching:** End of one time-slice & start of another.
* Multitasking creates an **illusion** that all tasks are running simultaneously, but in actual **frequent** context switching is occurring.

Advantages:-

* Quickly responsive interface
* Reduces idle time

Disadvantages:-

* Unsafe
* Expensive

**Multiprocessing OS**

* Multiple **processors** (**CPUs**) are present.
* It provides **tightly coupled system**.
* **Tightly coupled system:** Devices are **connected** to each other & **depend** on each other to work.
* Its opposite is **loosely coupled system**.

Advantages:-

* Increased throughput
* **Throughput:** Number of jobs executed per unit time.
* Economical (buying 3 CPU << buying 3 processors)
* Increased reliability

**Real Time OS (RTOS)**

* Hard real time OS
* Soft real time OS

**Distributed OS**

* Loosely coupled system
* Several computers share **same** OS.
* Computation is **distributed** among all processors & memory.

Advantages:-

* Shared resources
* Shared load
* Reliable
* Scalability

**Facilities by OS**

* Error detection
* Resource allocation
* Accounting
* Security & protection etc

**System Calls**

* System call is a **request** by the **user** to the **OS** for **accessing** a resource.
* This request can be for any hardware component.
* And this request is made using the kernel.

Kernal mode:-

* **Kernel:** Core of OS & controls almost everything going in the computer.
* In **kernel mode**, a user can access any hardware resources **directly**.
* So, kernel mode is believed to be very powerful mode.
* A failure in kernel may affect the **whole** system also.

User mode:-

* Outer layer of OS architecture.
* **No direct** access over the hardware.
* A failure in user mode might **not** affect the **whole** system.

**Process Management**

* **Process:** An executing program.

OS responsibilities with process management:-

* Process creation & deletion
* Process suspension & resumption
* Deadlock handling
* Provision for mechanisms
* Process synchronization
* Process communication

**Memory Management**

* **Memory:** Array of bytes with each byte having its own address.
* **Main memory:** RAM

OS responsibilities with memory management:-

* **Keep track** about memory usage.
* Decide **which** process to do & **when**.
* Allocate or deallocate memory space.

**File Management**

OS responsibilities with file management:-

* File creation & deletion.
* Directory creation & deletion.
* File manipulation facility.
* Copying & moving files etc.

**I/O Management**

OS responsibilities with I/O management:-

* Providing **buffer-caching** system.
* **Buffer:** Temporary storage to hold data before writing it permanently.
* **Caching:** Storage of data for quick access.
* **Buffer-caching:** **Buffer + Caching**
* Providing device driver interface.

**Secondary Storage Management**

* **Secondary storage:** Place where data is permanently stored.

OS responsibilities with process management:-

* Free space management
* Storage allocation
* Disk scheduling

**Protection**

OS responsibilities with protection:-

* Distinguish b/w **authorised** & **unauthorised** usage.
* Take proper security measures.

**Command-Interpreter System**

* A program that **interprets** commands.
* **CLI** in Windows, **Shell** in UNIX/Linux.

**Monolithic Kernel**

* A type of kernel.
* **System level** commands can be given.
* It can control **device drivers** from the **CLI**.

**Microkernel**

* A very **minimalist** type of kernel.
* Removes all the **unnecessary** part of the kernel.
* These removed parts are generally what could be done using **GUI**.
* Provided facility includes: **Memory management & communication facility etc.**
* Works on **message passing** principle.
* **Message passing:** A process indirectly passes its request, through other processes.

Advantages:-

* Takes less space.
* Easy to debug kernel.
* More secured & reliable.

Disadvantages:-

* Poor performance.
* Increased system overhead.
* **System overhead:** Large time taken due to heavy processing.

**Virtual Machine**

* ***\*Just recall what “Virtual Box” do…\****