OS is a system program that acts as a interface between user and hardware.

An operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information.

The kernel is a part of an operating system. The operating system is the software package that communicates directly to the hardware and our application. The kernel is the lowest level of the operating system. The kernel is the main part of the operating system and is responsible for translating the command into something that can be understood by the computer. The main functions of the kernel are:

1. memory management
2. network management
3. device driver IO management
4. file management
5. process management

An operating system also includes applications like the user interface (shell, gui, tools, and services).

[**Types of Operating System**](https://www.geeksforgeeks.org/operating-system-types-operating-systems-awaiting-author/)**–**

* Batch Operating System- Sequence of jobs in a program on a computer without manual interventions.
* Time sharing operating System- allows many users to share the computer resources.(Max utilization of the resources).
* Distributed operating System- Manages a group of different computers and make appear to be a single computer.
* Network operating system- computers running in different operating system can participate in common network (It is used for security purpose).
* Real time operating system – meant applications to fix the deadlines.

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

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

**2. Time-Sharing Operating Systems –**Each task has given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.

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

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

**Examples of Distributed Operating System are-** LOCUS etc.

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

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

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

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

**Multiprogramming** - A computer running more than one program at a time (like running Excel and Firefox simultaneously)

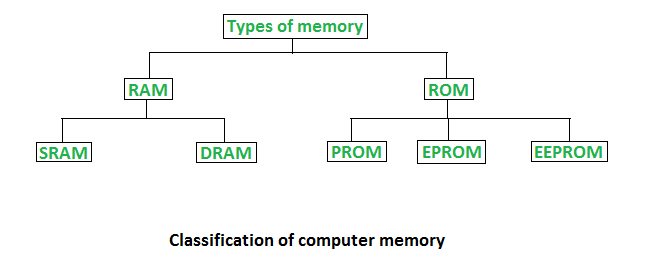
**Multiprocessing** - A computer using more than one CPU at a time

**Multitasking**: More than one task/program/job/process can reside into the same CPU at one point of time. This ability of the OS is called multitasking.

**Multiusers System** - a computer system in which multiple terminals connect to a host computer that handles processing tasks.

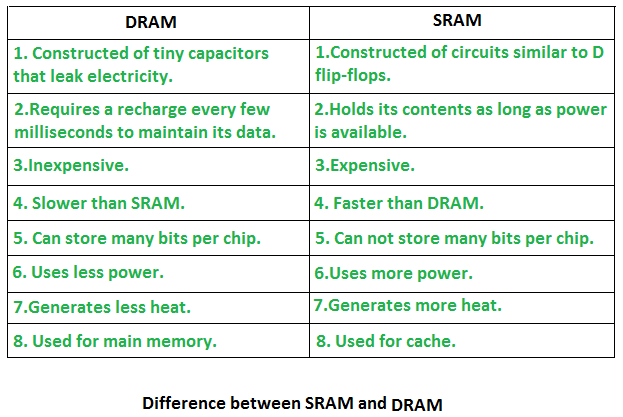
**Multithreading** - Multithreading is the extension of multitasking. Multithreading is the ability of an operating system to subdivide the specific operation within a single application into individual threads. Each of these threads can run in parallel. The OS divides processing time not only among different applications but also among each thread within an application.

Computer memory is of two basic type – Primary memory / Volatile memory and Secondary memory / non-volatile memory. Random Access Memory (RAM) is volatile memory and Read Only Memory (ROM) is non-volatile memory.



**1. Random Access Memory (RAM) –**

* It is also called as *read write memory* or the*main memory* or the *primary memory*.
* The programs and data that the CPU requires during execution of a program are stored in this memory.
* It is a volatile memory as the data loses when the power is turned off.
* RAM is further classified into two types- *SRAM (Static Random Access Memory)* and *DRAM (Dynamic Random Access Memory)*.

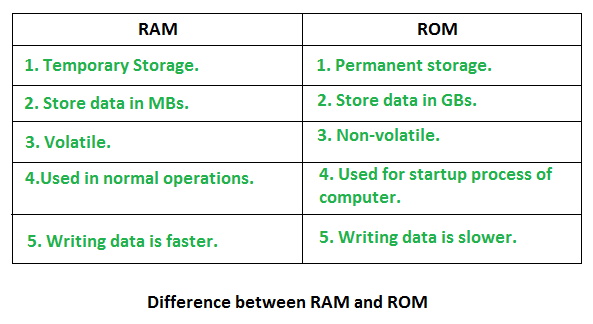


**2. Read Only Memory (ROM) –**

* Stores crucial information essential to operate the system, like the program essential to boot the computer.
* It is not volatile.
* Always retains its data.
* Used in embedded systems or where the programming needs no change.
* Used in calculators and peripheral devices.
* ROM is further classified into 4 types- *ROM*, *PROM*, *EPROM*, and *EEPROM*.

**Types of Read Only Memory (ROM) –**

1. **PROM (Programmable read-only memory)** – It can be programmed by user. Once programmed, the data and instructions in it cannot be changed.
2. **EPROM (Erasable Programmable read only memory)** – It can be reprogrammed. To erase data from it, expose it to ultra violet light. To reprogram it, erase all the previous data.
3. **EEPROM (Electrically erasable programmable read only memory)** – The data can be erased by applying electric field, no need of ultra violet light. We can erase only portions of the chip.



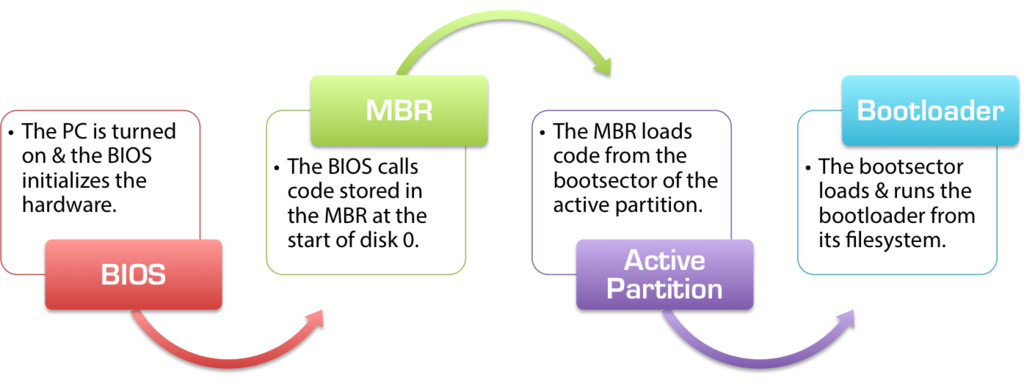
| **BASIS FOR COMPARISON** | **PRIMARY MEMORY** | **SECONDARY MEMORY** |
| --- | --- | --- |
| Basic | Primary memory is directly accessible by Processor/CPU. | Secondary memory is not directly accessible by CPU. |
| Altered Name | Main memory. | Auxiliary memory. |
| Data | Instructions or data to be currently executed are copied to main memory. | Data to be permanently stored is kept in secondary memory. |
| Volatility | Primary memory is usually volatile. | Secondary memory is non-volatile. |
| Formation | Primary memories are made of semiconductors. | Secondary memories are made of magnetic and optical material. |
| Access Speed | Accessing data from primary memory is faster. | Accessing data from secondary memory is slower. |
| Access | Primary memory is accessed by the data bus. | Secondary memory is accessed by input-output channels. |
| Size | The computer has a small primary memory. | The computer has a larger secondary memory. |
| Expense | Primary memory is costlier than secondary memory. | Secondary memory is cheaper than primary memory |
| Memory | Primary memory is an internal memory. | Secondary memory is an external memory. |

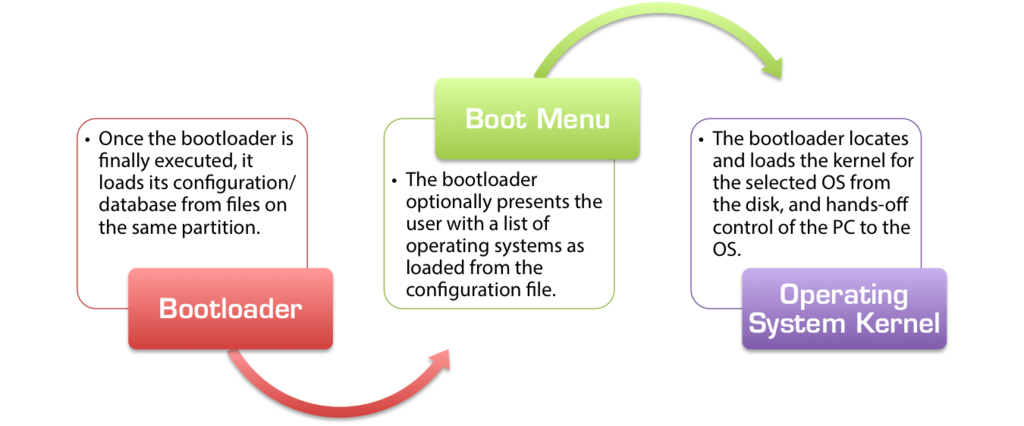
A boot loader, also called a boot manager, is a small program that places the [operating system](https://whatis.techtarget.com/definition/operating-system-OS) (OS) of a computer into [memory](https://searchstorage.techtarget.com/definition/memory-card). When a computer is powered-up or restarted, the basic input/output system ([BIOS](https://whatis.techtarget.com/definition/BIOS-basic-input-output-system)) performs some initial tests, and then transfers control to the [master boot record](https://whatis.techtarget.com/definition/Master-Boot-Record-MBR) (MBR) where the boot loader resides. Most new computers are shipped with boot loaders for some version of Microsoft Windows or the Mac OS. If a computer is to be used with [Linux](https://searchdatacenter.techtarget.com/definition/Linux-operating-system), a special boot loader must be installed.

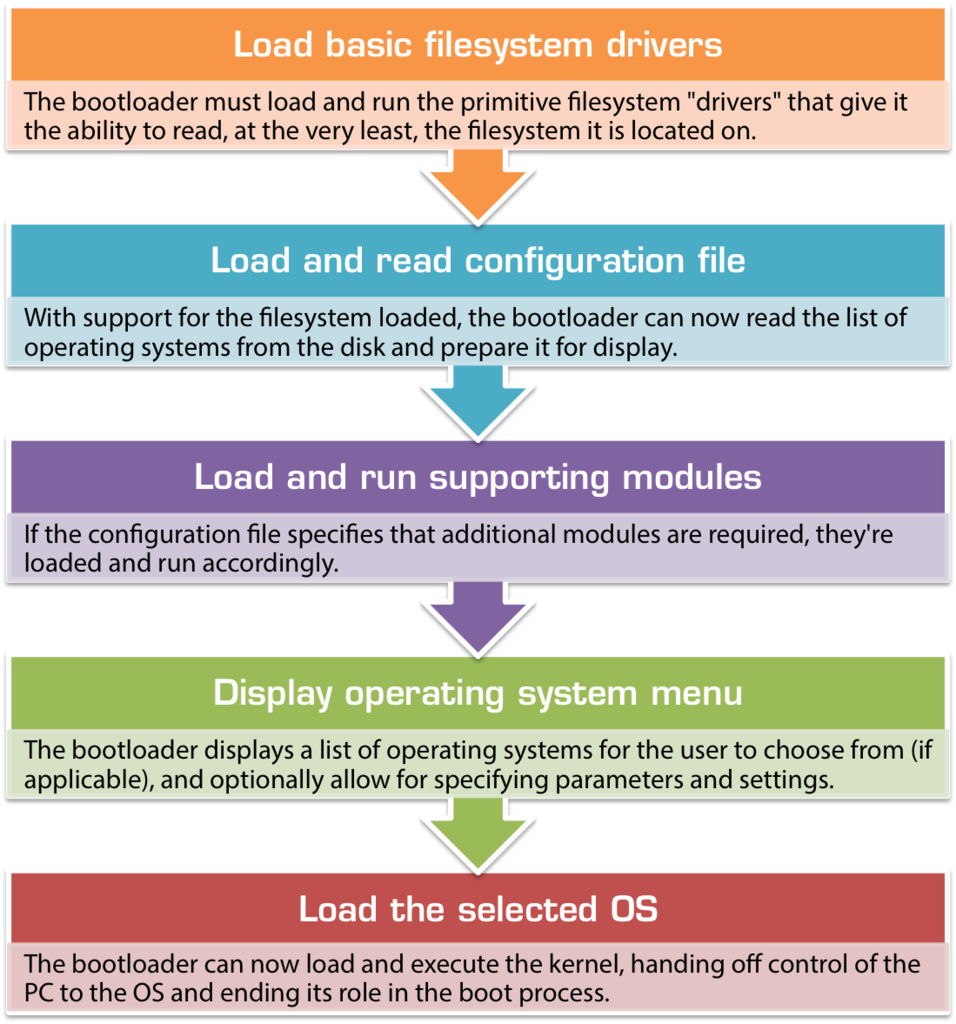
For Linux, the two most common boot loaders are known as [LILO](https://whatis.techtarget.com/definition/LILO-LInux-LOader) (LInux LOader) and LOADLIN (LOAD LINux). An alternative boot loader, called GRUB (GRand Unified Bootloader), is used with [Red Hat](https://searchdatacenter.techtarget.com/definition/Red-Hat) Linux.

Overview of the BIOS/MBR Boot Process

https://neosmart.net/wiki/mbr-boot-process/



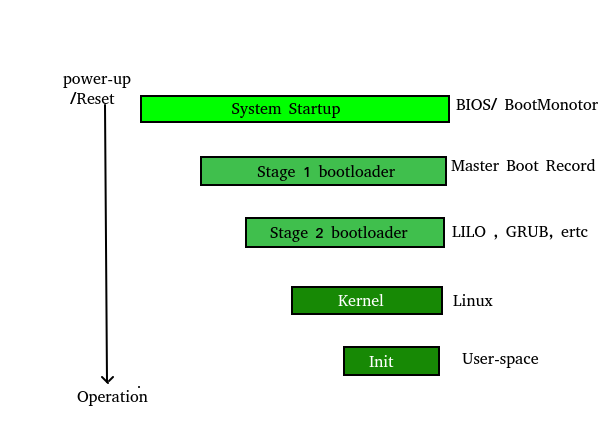




*A 32-bit system can access 232 memory addresses, i.e 4 GB of RAM or physical memory.  
A 64-bit system can access 264 memory addresses, i.e actually 18-Billion GB of RAM. In short, any amount of memory greater than 4 GB can be easily handled by it.*

So the maximum amount of RAM a processor can address is 2^(width of data bus in bits), given that the proper addressing mode is switched on in the processor.

A major difference between **32-bit processors and 64-bit processors** is the number of calculations per second they can perform, which affects the speed at which they can complete tasks. 64-bit processors can come in **dual core, quad core, six core, and eight core versions** for home computing.



The boot process is something that happens every time you turn your computer on.

**Functions of BIOS**

* POST(Power On Self Test) this test happens each time when turned on computer.
* It initializes the various hardware devices. It is an important process so as to ensure that all the devices operate smoothly without any conflicts. BIOSes following ACPI(Advanced Configuration and Power Interface) create tables describing the devices in the computer.
* The POST first checks the bios and then tests the CMOS RAM. If there is no problems with this then POST continues to check the CPU, hardware devices such as the Video Card, the secondary storage devices such as the Hard Drive, Floppy Drives, Zip Drive or CD/DVD Drives.If some errors found then an error message is displayed on screen or a number of beeps are heard. These beeps are known as POST beep codes.

**Master Boot Record**

* The Master Boot Record (MBR) is a small program that starts when the computer is booting, in order to find the operating system (eg. Windows XP). This complicated process (called the Boot Process) starts with the POST (Power On Self Test) and ends when the Bios searches for the MBR on the Hard Drive, which is generally located in the first sector, first head, first cylinder (cylinder 0, head 0, sector 1).
* The bootstrap loader is stored in the master boot record (MBR) on the computer’s hard drive. When the computer is turned on or restarted, it first performs the power-on self-test, also known as POST. If the POST is successful and no issues are found, the bootstrap loader will load the operating system for the computer into memory. The computer will then be able to quickly access, load, and run the operating system.

**init**  
  
init is the last step of the kernel boot sequence. It looks for the file /etc/inittab to see if there is an entry for initdefault. It is used to determine initial run-level of the system. A run-level is used to decide the initial state of the operating system.  
Some of the run levels are:

**Run Level**

 0 –> System Halt

 1 –> Single user mode

 3 –> Full multiuser mode with network

 5 –> Full multiuser mode with network and X display manager

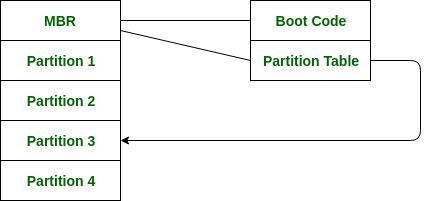
 6 –> Reboot

* The above design of init is called SysV- pronounced as [System five](https://en.wikipedia.org/wiki/UNIX_System_V). Several other implementations of init have been written now. Some of the popular implementatios are systemd and upstart.
* The next step of init is to start up various daemons that support networking and other services related to GUI.

computer bootstrap is stored in Read Only Memory (ROM).

1. This location is good for storage because this place doesn’t require initialization and moreover location here it is fixed so that processor can start executing when powered up or reset.
2. ROM is basically read-only memory and hence it cannot be affected by the computer virus.
3. ROM is non volatile type of memory.

The problem is that changing the bootstrap code basically requires changes in the ROM hardware chips.Because of this reason, most system nowadays has the tiny bootstrap loader program in the boot whose only job is to bring the full bootstrap program from the disk. Through this now we are able to change the full bootstrap program easily and the new version can be easily written onto the disk.



This code basically directs the system to read code directly from MBR. In addition to this boot code also contain the table which lists the partition for the hard disk and also a flag which basically indicates which partition is to be boot from the system.

**Booting Process With BIOS :** When BIOS begins it’s execution, it first goes for the Power-On Self Test (POST), which ensures that the hardware devices are functioning correctly. After that, it checks for the Master Boot Record in the first sector of the selected boot device. From the MBR, the location of the Boot-Loader is retrieved, which, after being loaded by BIOS into the computer’s RAM, loads the operating system into the main memory.

**Kernel** is the core part of an operating system which manages system resources. It also acts like a bridge between application and hardware of the computer. It is one of the first programs loaded on start-up (after the Bootloader).

* The operating system puts the CPU in kernel mode when it is executing in the kernel so, that kernel can execute some special operation.
* The operating system puts the CPU in user mode when a user program is in execution so, that user program cannot interface with the operating system program.

**Microkernel** Operating System remains unaffected as user services and kernel services are isolated so if any user service fails it does not affect kernel service. Thus it adds to one of the advantages in a microkernel. It is easily extendable i.e. if any new services are to be added they are added to user address space and hence requires no modification in kernel space. It is also portable, secure and reliable.

microkernel is solely responsible for the most important services of operating system they are named as follows:

* Inter process-Communication
* Memory Management
* CPU-Scheduling

**Advantages of Microkernel –**

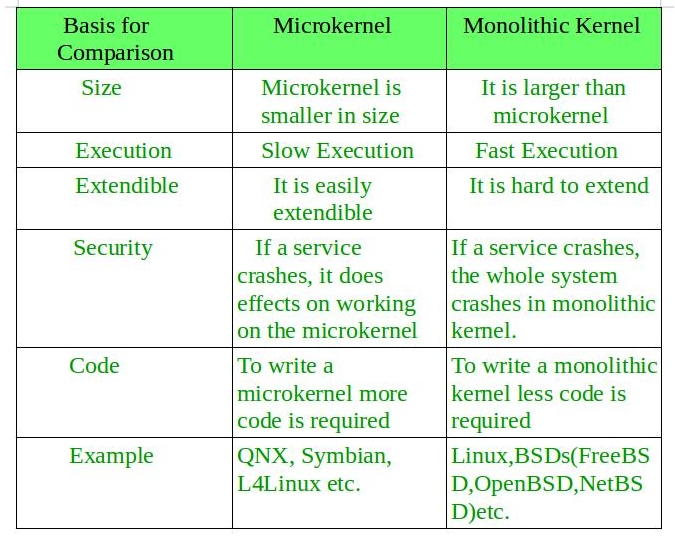
* The architecture of this kernel is small and isolated hence it can function better.
* Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

**Eclipse IDE** is a good example of Microkernel Architecture

**Monolithic Kernel** is another classification of Kernel. Like microkernel this one also manages system resources between application and hardware, but **user services** and **kernel services** are implemented under same address space. It increases the size of the kernel, thus increases size of operating system as well.

As both services are implemented under same address space, this makes operating system execution faster.

If any service fails the entire system crashes, and it is one of the drawbacks of this kernel. The entire operating system needs modification if user adds a new service.



 A system call is a way for application programs to interact with the operating system kernel.

System call provides the services of the operating system to the user programs via Application Program Interface(API).

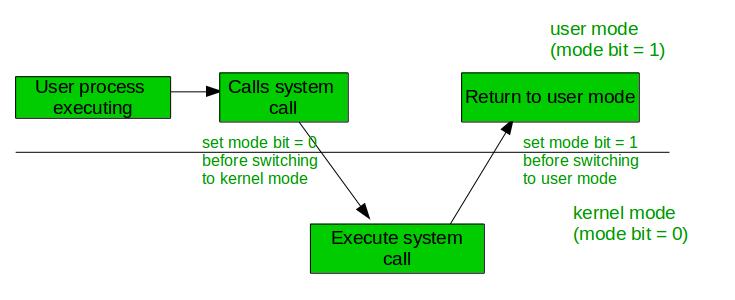
**Examples of Windows and Unix System Calls**

|  |  |  |
| --- | --- | --- |
|  | **WINDOWS** | **UNIX** |
| Process Control | CreateProcess() ExitProcess() WaitForSingleObject() | fork() exit() wait() |
| File Manipulation | CreateFile() ReadFile() WriteFile() CloseHandle() | open() read() write() close() |
| Device Manipulation | SetConsoleMode() ReadConsole() WriteConsole() | ioctl() read() write() |
| Information Maintenance | GetCurrentProcessID() SetTimer() Sleep() | getpid() alarm() sleep() |
| Communication | CreatePipe() CreateFileMapping() MapViewOfFile() | pipe() shmget() mmap() |
|  |  |  |
| Protection | SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup() | chmod() umask() chown() |

*Each resource has an associated****soft and hard limit.***

* ***soft limit:****The soft limit is the actual limit enforced by the kernel for the corresponding resource.*
* ***hard limit:****The hard limit acts as a ceiling for the soft limit.*

***The soft limit ranges in between 0 and hard limit.***



*The Instructions that can run only in Kernel Mode are called Privileged Instructions .*

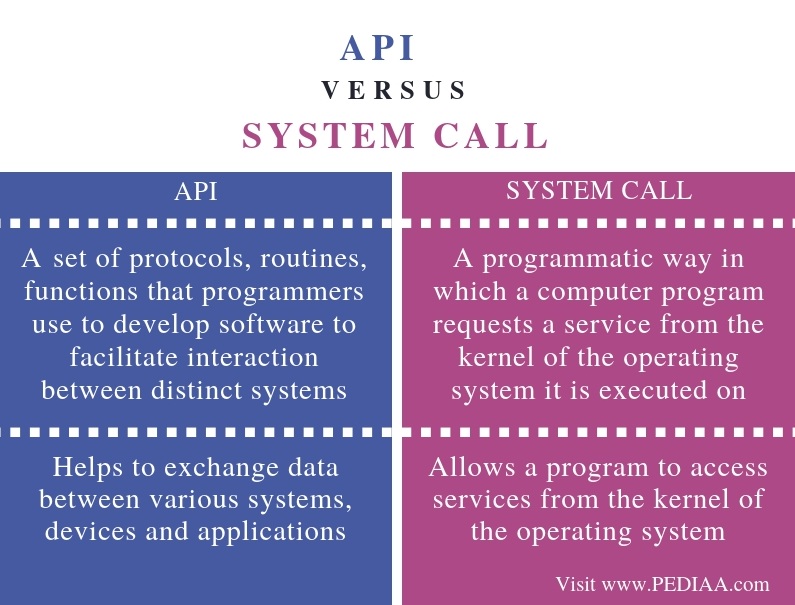
examples of Privileged Instructions include:

* I/O instructions and Halt instructions
* Turn off all Interrupts
* Set the Timer
* Context Switching
* Clear the Memory or Remove a process from the Memory
* Modify entries in Device-status table

*The Instructions that can run only in User Mode are called Non-Privileged Instructions .*

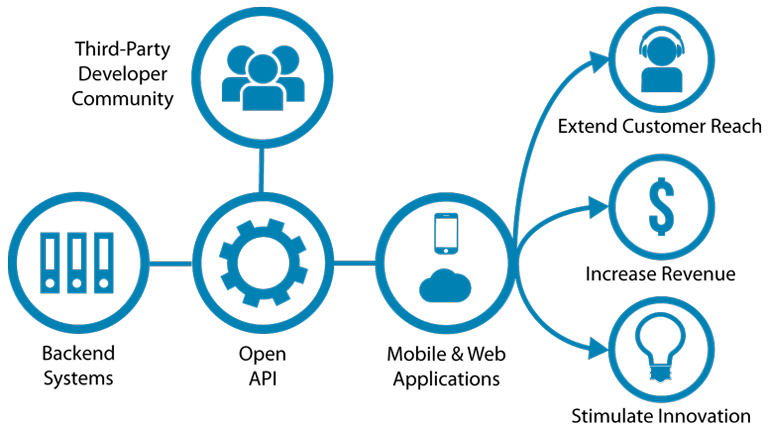
examples of Non-Privileged Instructions include:

* Reading the staus of Processor
* Reading the System Time
* Generate any Trap Instruction
* Sending the final prinout of Printer



What is an API

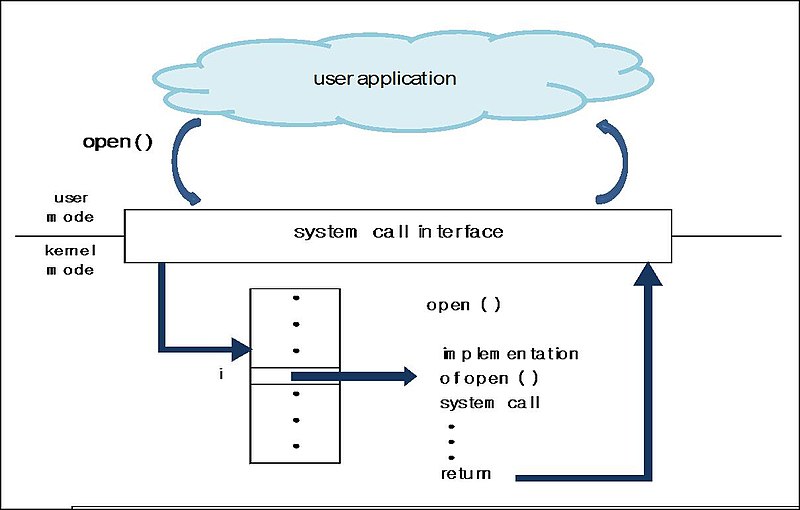
Different devices and applications share data between them. Some of them include online reservations and booking systems. API (**Application Programming Interface**) helps to establish connectivity among devices and applications. Moreover, it is an interface that takes the requests from the user and informs the system about what should be done and return the response back to the user.



For example, assume an online travel service that aggregates information from multiple airlines. The travel service interacts with the airline’s API. The API takes the requests to book seats and select meals from the travel service to the airline system. Then it delivers the airlines responses back to the online travel service and the travel service displays the details to the users. This is a real-world application for an API.

What is a System Call

A computer operates in two modes: user mode and kernel mode. The execution of a program is in the user mode. When the program requires [RAM](http://pediaa.com/difference-between-cache-and-ram/#RAM) or a hardware resource, it sends a request to the kernel and the mode changes from user mode to kernel mode. System calls refer to these requests. After completing the task, the mode changes back to the user mode. Likewise, the mode changes continuously. This transition is also called context switching.



e.g – open(),read(),write(),ioctl()

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

**Attributes or Characteristics of a Process**

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

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

**3. CPU registers:** Like the Program Counter (CPU registers must be saved and

restored when a process is swapped in and out of CPU)

**5. Accounts information:**

**6. I/O status information:** For example, devices allocated to the process,

open files, etc

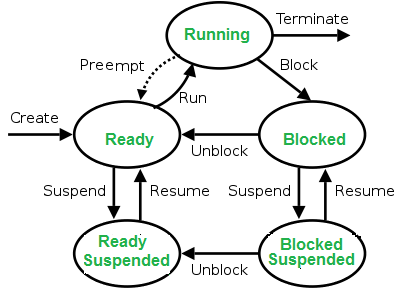
**8. CPU scheduling information:** For example, Priority (Different processes

may have different priorities, for example

a short process may be assigned a low priority

in the shortest job first scheduling)

All of the above attributes of a process are also known as the ***context of the process***.  
Every process has its own [program control block](http://en.wikipedia.org/wiki/Process_control_block)(PCB), i.e each process will have a unique PCB. All of the above attributes are part of the PCB.



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

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

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

While creating a process the operating system performs several operations. To identify these process, it must identify each process, hence it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process’s execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc. All these information is required and must be saved when the process is switched from one state to another. When the process made transitions from one state to another, the operating system must update information in the process’s PCB.

There are three types of process scheduler.

* 1. Long term or Job scheduler: Brings jobs from New state to Ready state by making a careful selection of both I/O and CPU bound process.
  2. Short term or CPU scheduler: It is responsible for selecting one process from ready state for scheduling it on the running state.

***Dispatcher*** is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:

i)Switching context.

ii)Switching to user mode.

iii)Jumping to the proper location in the newly loaded program.

* 1. **Medium-term scheduler** It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa).

**Why do we need scheduling?**  
A typical process involves both I/O time and CPU time. In a uniprogramming system like MS-DOS, time spent waiting for I/O is wasted and CPU is free during this time. In multiprogramming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

**Objectives of Process Scheduling Algorithm**

Max CPU utilization [Keep CPU as busy as possible]

Fair allocation of CPU.

Max throughput [Number of processes that complete their execution per time unit]

Min turnaround time [Time taken by a process to finish execution]

Min waiting time [Time a process waits in ready queue]

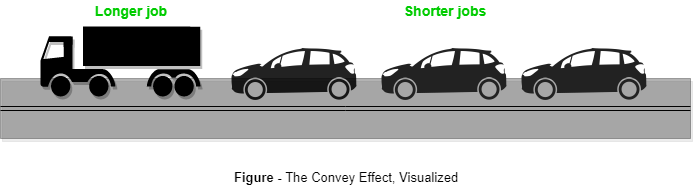
Min response time [Time when a process produces first response]

**Different Scheduling Algorithms**

***First Come First Serve (FCFS):*** Simplest scheduling algorithm that schedules according to arrival times of processes. First come first serve scheduling algorithm process that requests the CPU first is allocated the CPU first. It is implemented by using the FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. FCFS is a non-preemptive scheduling algorithm.

**Note:**First come first serve suffers from [convoy effect](https://www.geeksforgeeks.org/convoy-effect-operating-systems/).

Convoy Effect is phenomenon associated with the First Come First Serve (FCFS) algorithm, in which the whole Operating System slows down due to few slow processes.



FCFS algorithm is non-preemptive in nature, that is, once CPU time has been allocated to a process, other processes can get CPU time only after the current process has finished. This property of FCFS scheduling leads to the situation called Convoy Effect.

Convoy Effect is phenomenon associated with the First Come First Serve (FCFS) algorithm, in which the whole Operating System slows down due to few slow processes.

***Shortest Job First(SJF):*** Process which have the shortest burst time are scheduled first.If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.

***Longest Job First(LJF):*** It is similar to SJF scheduling algorithm. But, in this scheduling algorithm, we give priority to the process having the longest burst time. This is non-preemptive in nature i.e., when any process starts executing, can’t be interrupted before complete execution.

***Shortest Remaining Time First(SRTF):*** It is preemptive mode of SJF algorithm in which jobs are schedule according to shortest remaining time.

***Longest Remaining Time First(LRTF):*** It is preemptive mode of LJF algorithm in which we give priority to the process having largest burst time remaining.

***Round Robin Scheduling:*** Each process is assigned a fixed time(Time Quantum/Time Slice) in cyclic way.It is designed especially for the time-sharing system. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum. To implement Round Robin scheduling, we keep the ready queue as a FIFO queue o£ processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1-time quantum, and dispatches the process. One of two things will then happen. The process may have a CPU burst of less than 1-time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1-time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail o£ the ready queue. The CPU scheduler will then select the next process in the ready queue.

***Priority Based scheduling (Non-Preemptive):*** In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then schedule according to arrival time. Here starvation of process is possible.

***Highest Response Ratio Next (HRRN)*** In this scheduling, processes with highest response ratio is scheduled. This algorithm avoids starvation.

Response Ratio = (Waiting Time + Burst time) / Burst time

***Multilevel Queue Scheduling:***According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue. Only after completion of processes from top level queue, lower level queued processes are scheduled. It can suffer from starvation.

***Multi level Feedback Queue Scheduling:***It allows the process to move in between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it is moved to a lower-priority queue.