1 Introduction To Operating System

1.1 OPERATING SYSTEM FUNDAMENTALS

Introduction:

Definition

"An operating system is a software which acts as a mediator/interface in between the user and computer hardware." In other words, an OS is a software that performs work such as memory management, process management, file management, etc.

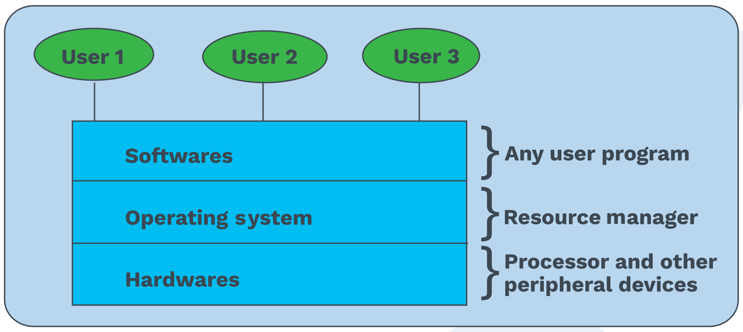


Fig. 1.1 Basic Overlay of Computer System

Operating System acts as Resource Manager/Allocator because it takes care of the responsibility to allocate the CPU to some process and I/O to some other process.
Software/Applications communicate with the hardware through Operating System (O.S.) using system calls. For different kinds of services, there exist different system calls.

Objectives of an operating system:
Mainly two objectives: Primary and Secondary.

1) **Primary objective: Convenience**
The main aim of an operating system is to give convenience to the user. Users can perform work like process scheduling, and conversion of user code to machine code easily and conveniently with the help of OS.
2) **Secondary objective: Efficiency**
It is a task of the operating system to manage the resources in such a way that resources are not kept to be idle and can be utilized efficiently.

Note:

The primary and secondary objective of an operating system is not the same for all operating systems.

- It can vary depending on the purpose of the operating system.

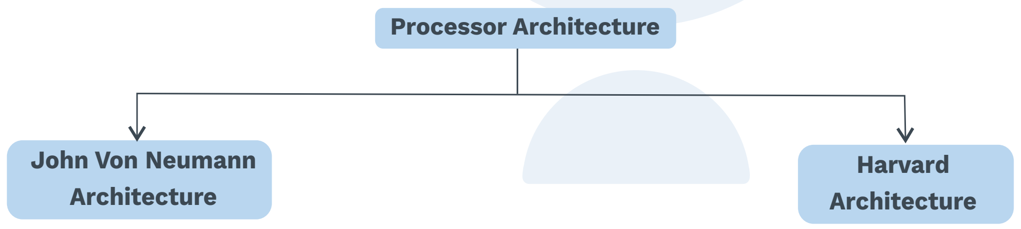
For example:

- The primary goal of WINDOWS → Convenience
- The primary goal of LINUX → Efficiency

Note:

Design and principles of operating systems directly depend on the computer architecture.

Types of Architecture



John von neumann architecture:

- John Von Neumann is considered one of the greatest mathematicians and computer scientists who came up with a simple and elegant architecture of what a computer is. Neumann architecture is also known as stored-program concept. The concept of a stored program states that any program we want to execute is going to be stored in RAM.
- A program is a sequence of instructions that are there in machine level language. The CPU reads these instructions from memory and executes them one by one in a sequence.
- In this architecture, physical memory is used to store instructions and data both.
- There is a common bus for data and instruction transfer.
- Instructions usually take 4 or more clock cycles to execute.
- Cost-wise cheaper.
- Instructions, as well as read/write, cannot be accessed by the CPU at the same time.
- In personal and small computers, this structure is used.

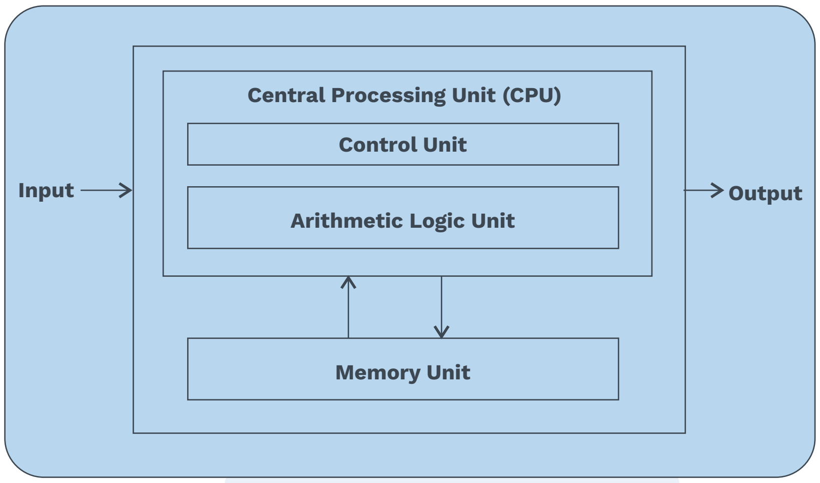


Fig. 1.2 Central Processing Unit (CPU)

Note:

$$\begin{tikzcd}\boxed{\text{C program}} \arrow[r, "\text{Compiled}"] & \boxed{\text{Machine Level Language}}\end{tikzcd}$$

Machine Level Language: The sequence of instructions specific to a microprocessor is stored in RAM.

Harvard Architecture:

In Von Neumann architecture, same memory is used to store instructions and data and a common bus is multiplexed to read/write instructions and data. The CPU is bound to read the instruction and read/write data alternatively in Von Neumann architecture. In Harvard Architecture separate buses are used for instruction and data. It is considered an advancement of Von Neumann Architecture. The CPU can read the instruction and read/write data in parallel in Harvard architecture.

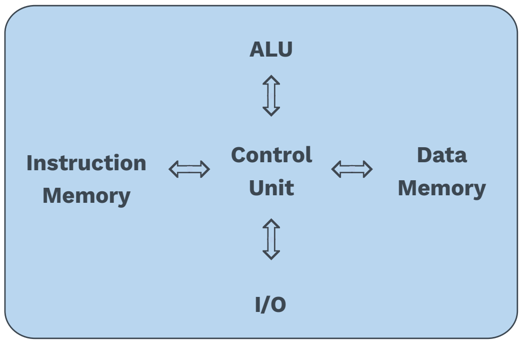
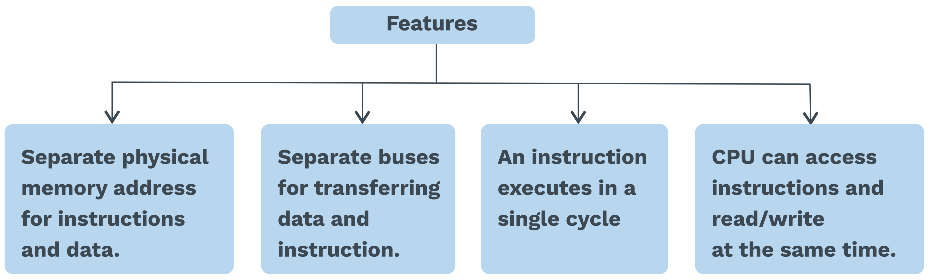


Fig. 1.3 Harvard Model


Features
Separate physical memory address for instructions and data.
Separate buses for transferring data and instruction.
An instruction executes in a single cycle
CPU can access instructions and read/write at the same time.

Note:

- Harvard architecture is more costly than John Von Neumann architecture
- Harvard architecture is used in microcontrollers and signal processing.

History of operating system:

1) First Generation (1940's -1950's):

In that era, punch cards were used. There was no operating system. Computers were of big size.

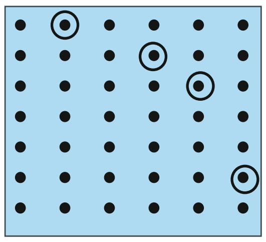


Fig. 1.4 Punch Card

Each line would represent some set of instructions. They would punch a hole and use electron techniques to read these codes from punch cards and convert them into a program. The magnetic Drum was the form of main memory.

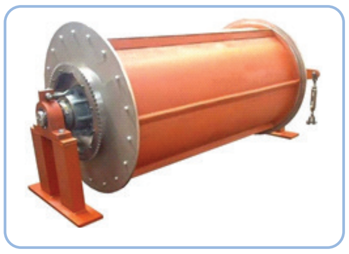


Fig. 1.5 Magnetic Drum

2) **Second generation (1950 – 1970):**
Still no operating system, people started using magnetic tapes as the permanent way to store data, e.g: Cassette tapes etc.
Data is moved from these tapes to RAM and then processed. So it can store more programs compared to First Generation Punch Cards. This era was called Batch Processing Era.

3) **Third generation (1980 – 1990):**
Operating Systems started to boom, and also disk technology became popular. These were early stages of First Hand Operating Systems like MS-DOS, Unix etc., which were built in this generation. Ram size has drastically increased and hard disk was introduced Multiprogramming also became popular.

4) **Fourth generation (2000 – present):**
New function specific Operating Systems were built like Network Operating Systems. Also called distributed Operating Systems, where we have hundreds of computers connected in the network. Real Time Operating Systems etc. were made and became popular for their specific purposes.

Functions of an operating system

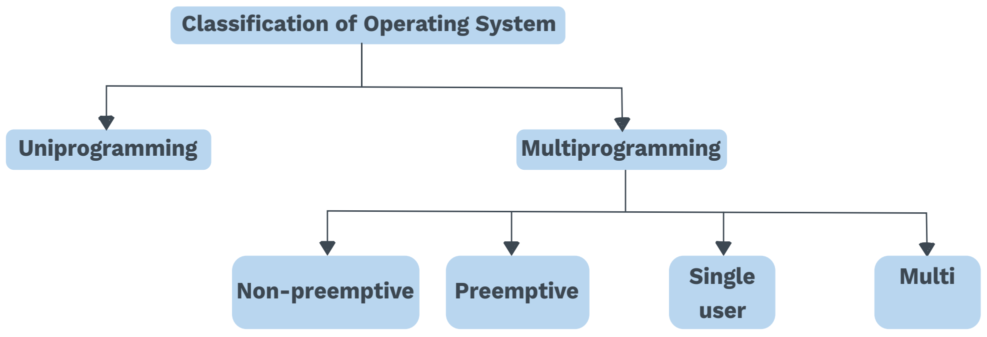
1) **Process management:**
   - A situation when the ready queue has multiple processes, but the processor can process only one process at a time.
   - In such a situation, the CPU has to schedule processes one at a time such that every process gets a fair chance to execute.
   - It means the CPU should not give priority to one particular process.
     Some well-known CPU scheduling algorithms:
   - FCFS(First Come First Serve), RR(Round Robin), etc.

2) **Memory management:**
For any process execution, it needs to be placed in memory first. The memory gets free after the completion of process execution, and other processes can use that memory. OS allocates and deallocates the memory for the process resulting in efficient use of memory.

3) **I/O Device management:**
   - While executing a process, the process needs access to different I/O devices.
   - But a process can not access any of the I/O devices directly. It requires to take the help of OS to access these devices.

4) **File management:**
A file system residing within the operating system manages different kinds of files, folders(directories) within the computer.

Classification of operating system:



Uniprogramming:

- In uniprogramming system can execute only one program at once.
- A lot of Wastage of CPU resources occurs here.
- We were using uniprogramming in old computers and mobiles.

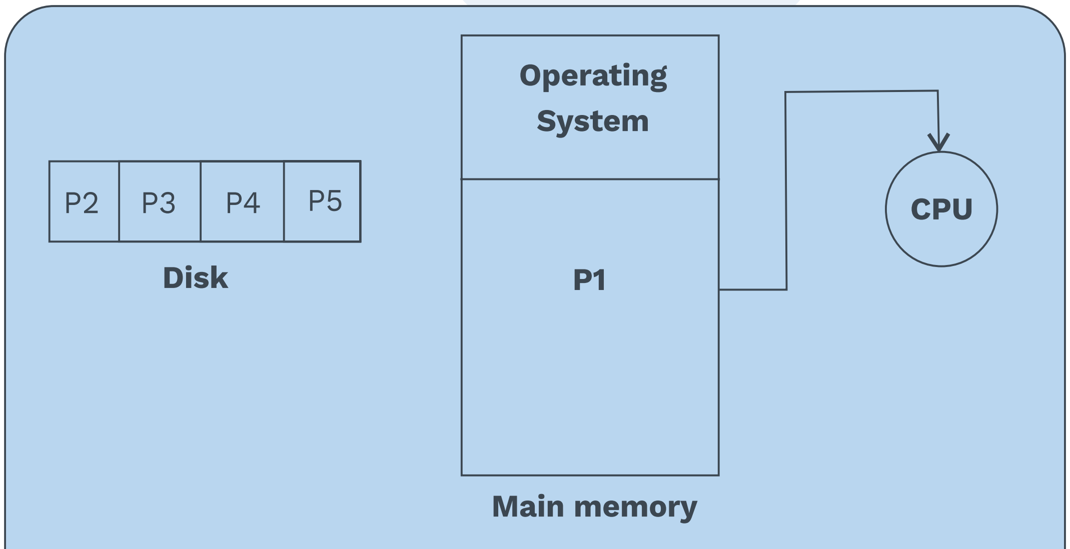


Fig. 1.6 Uniprogramming Memory Management Technique

Example:

1) Batch handling in old computers and mobiles.
2) Old portable working system.
3) Some operating systems which is Uniprogramming are Batch operating system, MS-DOS etc.

Multiprogramming:

- Multiprogramming came into the picture to overcome the shortcomings of Uniprogramming.
- Operating system is able to manage multiple ready to run programs in memory at once.

Note:

Context Switching:

It is the procedure of storing the state of a process or thread to be restored and resume execution later, so all programs are given a suitable amount of time.

In a multiprogramming environment processor needs to run various process, so processes need context switching from CPU to memory frequently, because only one process can run on the CPU at a time.

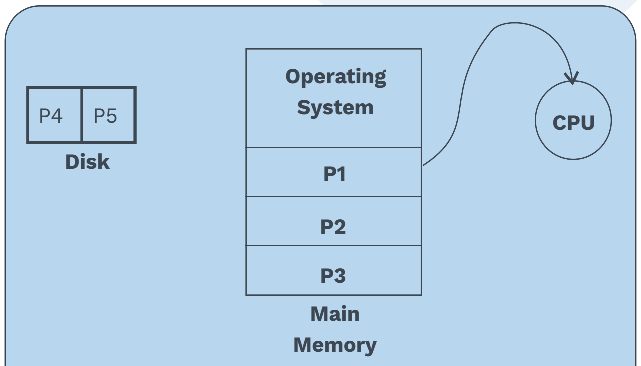


Fig. 1.7 Multiprogramming Memory Management Technique

Example:

Multiprogramming, Multitasking, Real-Time and Modern Operating Systems like Windows 7, 8, 10.

Types of multiprogramming:

1) Non-Preemptive based:

In this case, the process currently running releases the CPU voluntarily: The process releases CPU:

i) When they require I/O

ii) When the process invokes the system call

iii) When the process gets completed.

Drawbacks:

a) Starvation **b)** Less Response time overall

e.g., First Version of Windows, i.e. Windows 3.0, 3.11 etc.

2) Preemptive based:

It allows forceful reallocation of the CPU from the current running process to another process from the process pool.

When the running process can be preempted?

i) When I/O required

ii) Time quantum exhausts (RR Scheduling)

ii) High priority process comes

3) Single user based:

A Single user does all the programs submitted to the main memory for an execution. E.g., Windows XP, MS-DOS, Palm OS, etc.

4) Multi user based:

Programs from multiple users are submitted to the main memory for an execution.

E.g., Unix, Linux, Web-server based etc.

Architecture support for multiprogramming:

1) Address translation (memory):

The process of translating Logical Address(Disk Address) to Physical Address (Main Memory Address)

2) DMA (direct memory access)/IO device:

Independent to the CPU

Simultaneous working with CPU.

3) Two modes of CPU execution:

User Mode

Kernel Mode

Operating system types:

Various operating systems exist for various purposes.

Batch operating system:

In a batch operating system, similar jobs get grouped into batches with the help of some operator, and these batches get executed one by one. For example, let us assume that we have two programs that need to be executed and need Input-Output devices. While running, when a process leaves the CPU for the Input-Output device. It cannot load another program into the CPU until the currently running process terminates successfully.

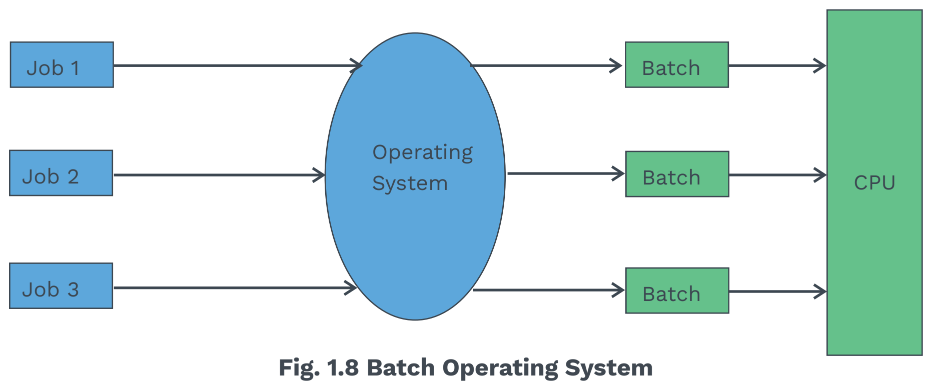


Fig. 1.8 Batch Operating System

- Batch OS is not good for interactive applications.
- In this OS, there is an operator that takes similar jobs together and groups them into batches.

Examples of batch os:
Payroll System, Bank Statement, etc.

Advantages:
i) Repeated jobs are done fast in batch systems without user interaction.
Disadvantages:
i) Increased CPU idleness.
ii) Decreased throughput of the system.

Note:
Throughput: Number of jobs completed per unit time.

Multiprogramming Operating System:
In today's scenario, a system has to execute various applications simultaneously. The operating system has to provide the required resources to all the applications for efficient and effective execution. The main memory accommodates several ready to execute jobs, and the operating system schedules them one after another on the CPU for execution.
In this multiprogramming environment, when an executing process needs I/O, it gets blocked and waits in I/O queue, by the time the operating system schedules other processes from the ready queue to execute on the CPU. The operating system does the scheduling processes repeatedly.

Advantages:
i) Throughput of system increases.
ii) Processor idle time is less. Despite a process waiting for the I/O event to happen; meanwhile, it keeps busy with another process.

Disadvantages:

i) Requires efficient memory management.

ii) Requires CPU Scheduling

iii) Tracking of all processes is sometimes complex.

Multitasking operating system:

Multitasking is considered an advancement to multiprogramming. A multitasking operating system simultaneously executes multiple tasks residing in the main memory. When a system has multiple processors, and they can be used to execute multiple instructions parallelly then it is known as multiprocessing. Hence multiple processes are required for multitasking while a single processor is needed to do multitasking.

In a modern computer, many tasks like text editor, web browser, mp3 player, etc can be executed simultaneously by a user. The multitasking operating system switches these tasks very fast, which users can't notice and get a feel of parallel execution of these tasks.

The only difference between multitasking and multiprogramming is that the later works on the concept of context switching, whereas the former works on time-sharing alongside context switching.

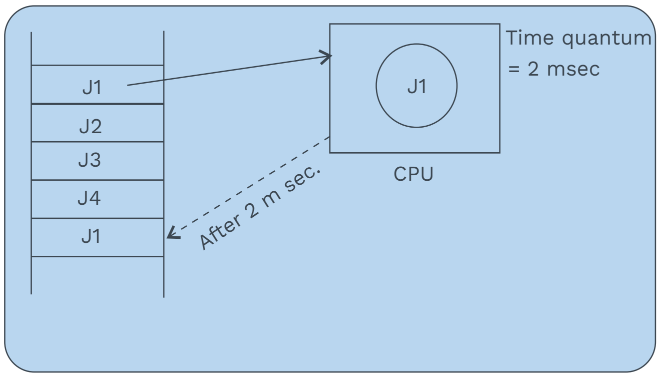


Fig. 1.9 Multitasking Operating System

Each job has a time constraint with it only inside which job can remain inside CPU. So, each job gets its turn so fast it looks as if every process is executing simultaneously.

Real-time operating system

Definition

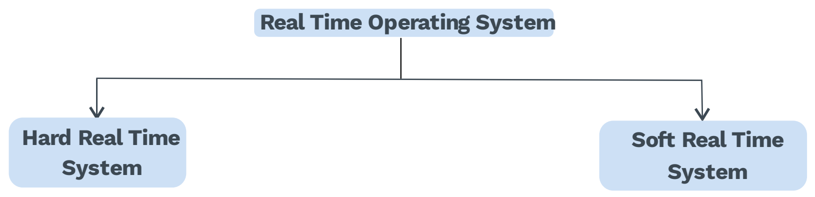
An OS that processes data as it comes in without any buffer delays.

It works on strict deadline. Response must be given within the deadline.

Examples: Traffic Control Systems, Missile Systems, etc.

Note:

The time interval required to process and respond to inputs is known as response time and it needs to be minimum.



Hard real time system:
An OS that works for those applications whose deadlines are very strict.
Example: Satellite control

Soft real time system:
These OS works for applications whose deadlines are narrow(less strict).
Real Time OS works mainly on primary memory only.
Example: Banking

Advantages of real time os:
Error free
Gives more focus on running applications.
Utilization of devices occurs in best possible way.

Disadvantages/Restrictions of Real Time OS:
1) Limited task
2) Use heavy system resources
3) Complex Algorithm
4) Thread priority

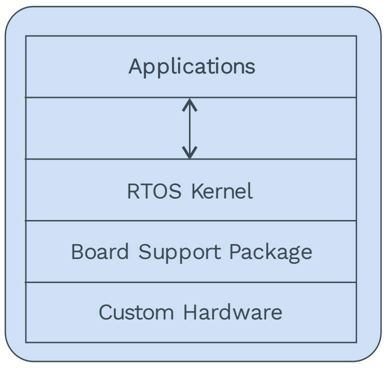


Fig. 1.10 Block Diagram of RTOS

Rack Your Brain

Can we use multitasking OS in airplane control systems?

Distributed OS:

Definition

"A distributed operating system is system software over a collection of independent, networked, communicating, and physically separate computational nodes. The puspose of this OS is to tackle the jobs that are executed by many CPUs."

In the distributed system(loosely coupled systems), there is a shared communication-network through which different interconnected computers can communicate.
Where all the individual systems have their own CPU and memory unit.
The function and size of the processor of these systems are different.
Benefit: Any user can access the files that are present in any other connected system in the network but absent in users own system.

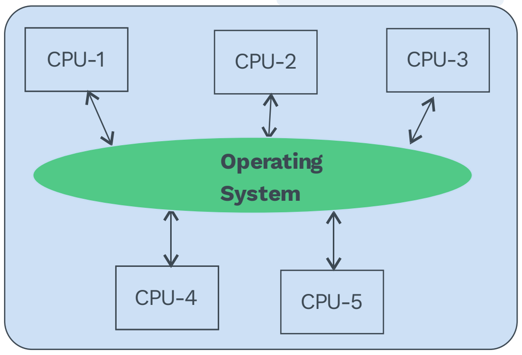


Fig. 1.11 Block Diagram of Distributed Operating System

Advantages:
1) No single point of failure
2) Efficiency and Interoperability
3) More room for Innovation

Disadvantages:
1) More things to secure
2) Complex System Design
3) Monitoring / Security

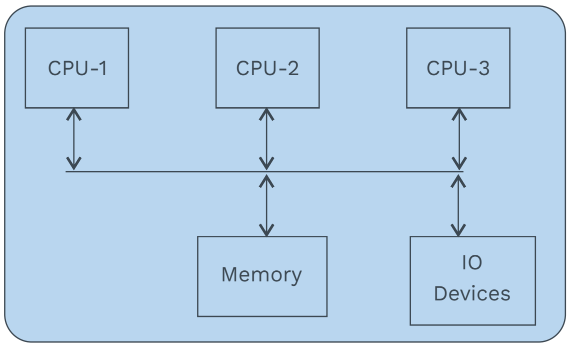


Fig. 1.12 Block Diagram of Multi Processor Systems

Multi processor systems

A single computer system with more than one processor uses an operating system called multiprocessor operating system. In this system the multiple processors share common computer bus, memory, peripheral devices, etc.
eg.: Unix

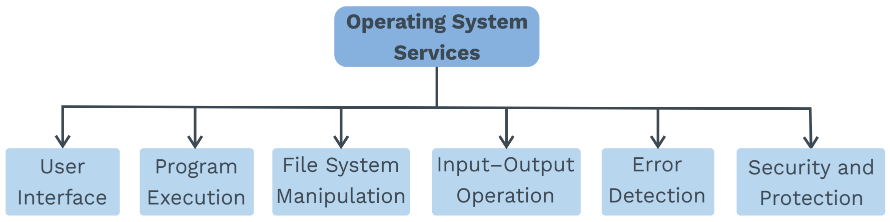
Advantages:
1) Throughput of the system will enhance.
2) Reliability of the system becomes high

Disadvantages:
1) Increased Expense
2) Complicated Operating System code
3) Large Main Memory Required

Services of OS:

OS works as an interface between user and hardware that provides some services to the user to implement the program.
OS services is not same for every OS, it changes from one OS to another.
All these services are given by OS to make the programming easy.

These services are as follows:



1) **User interface:**
   The operating system is designed in a such a way that it provides a user interface which is easy to use by a novice user.
   Example: Batch Interface, GUI(Graphical User Interface)
2) **Program execution:**
   OS provides the feature to load the program into a memory and to execute it.
3) **File-system manipulation:**
   OS provides feature like read/write a file,creation and deletion of the files, etc to the programmer.
4) **Input-output operations:**
   In order to maintain the protection, the I/O devices cannot be controlled by the users directly, OS helps user to access these devices.
5) **Error detection:**
   OS is responsible to handle any kind of errors whether it is present in hardware, I/O devices, within network.
   For all kind of error, a proper action needs to be taken by an OS.
6) **Protection and security:**
   Sometimes in a system where multiusers are present, OS plays an important role to ensure security and protection by controlling the user or process to access only those resources and information that are required for them.

Booting of Operating System:
All the components of a hardware as well as OS have to work properly for a computer system to start(boot).
It will result into a failed boot sequence if any of these elements fail.

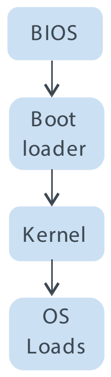
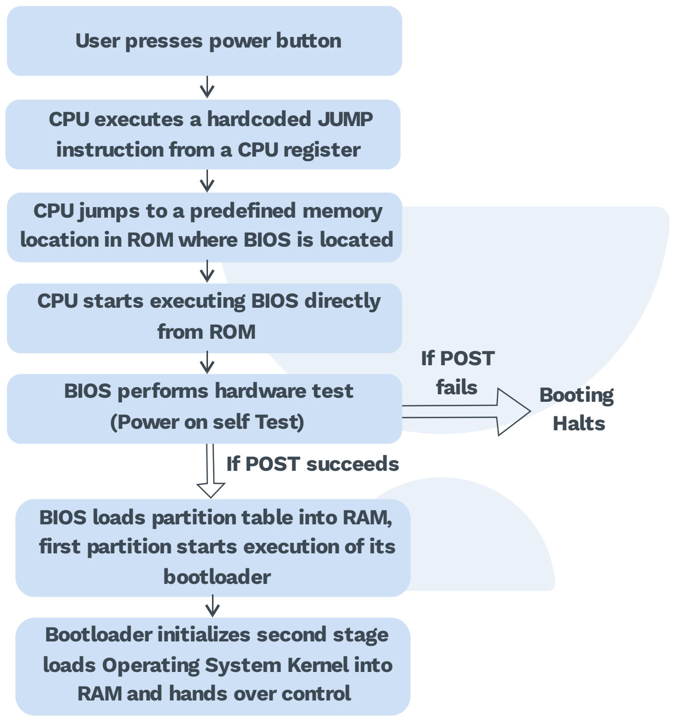


Fig. 1.13 Steps involved in a Boot Process

Booting is a startup sequence that starts the Operating System of a computer when it is turned on.

- When the computer is switched on, the boot program is the first to get executed, and it starts the OS in the computer.
- Boot sequence is available in every computer.
- Kernel is located by the bootstrap loader which loads the kernal into the main memory, and then the execution starts.



User presses power button
CPU executes a hardcoded JUMP instruction from a CPU register
CPU jumps to a predefined memory location in ROM where BIOS is located
CPU starts executing BIOS directly from ROM
BIOS performs hardware test (Power on self Test)
If POST fails
Booting Halts
If POST succeeds
BIOS loads partition table into RAM, first partition starts execution of its bootloader
Bootloader initializes second stage loads Operating System Kernel into RAM and hands over control

Fig. 1.14 Computer Booting Sequence

Rack Your Brain

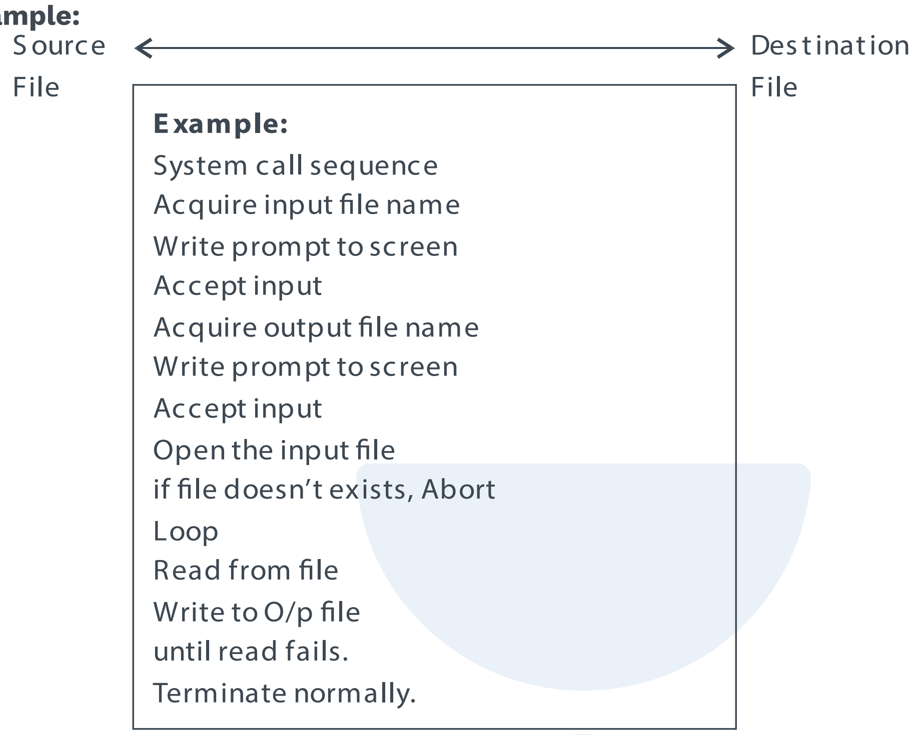
How booting process will change if there are more than one OS installed in computer system?

System Calls:

Definition

System call is a request for the kernel to access a resource.
It gives an interface to the services provided by the OS.
These are typically written in language C, C++.

Example:



```
Example:
System call sequence
Acquire input file name
Write prompt to screen
Accept input
Acquire output file name
Write prompt to screen
Accept input
Open the input file
if file doesn't exists, Abort
Loop
Read from file
Write to O/p file
until read fails.
Terminate normally.
```

Fig. 1.15 Example of How System Calls are Used

In the above figure, even a small operation of copying a file into another file takes many sequences of system calls to the hardware. Most users never see this level of detail because application developers design programs according to an API (Application Programming Interface).

Application programming interface(API):

Definition

"The API specifies a set of functions that are available to an application programmer, including the parameteres that are passed to each function and the return values the programmer can expect."

Handling of system call:

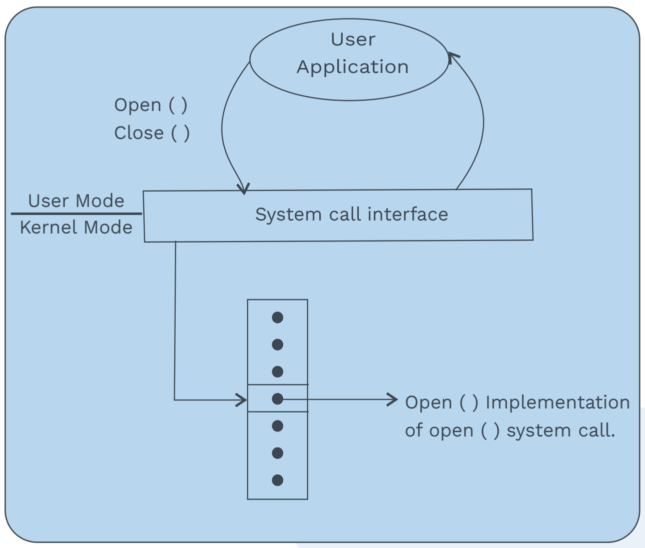


Fig. 1.16 Open ( ) System Call Handlings

1) Usually, for any programming language, an interface to system call is given by the runtime support system.
2) Each system call is assigned a number and an indexed table is managed by the system call-interface.
   A system call is then invoked in the kernel and the status of the system call is returned by the interface
3) User need not worry about how the system call is working, the caller of the system call just has to follow the API and see the result after the execution of the system call.
4) Methods for passing parameters to the OS.

i)	Registers	—	less size parameter
ii)	Block or table	—	address of the block is passed
iii)	Stack method	—	This does not limit the number of parameters being passed.

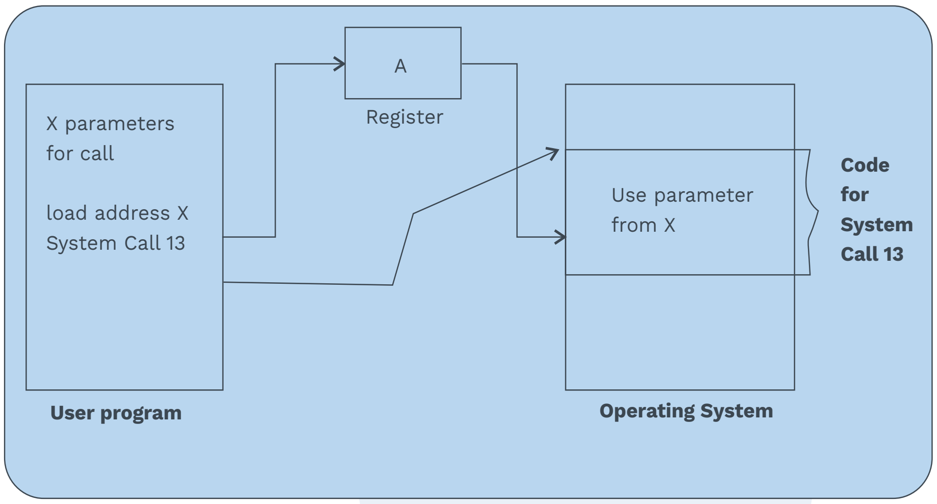


Fig. 1.17 Passing of Parameters as a Table

Types of System Calls:

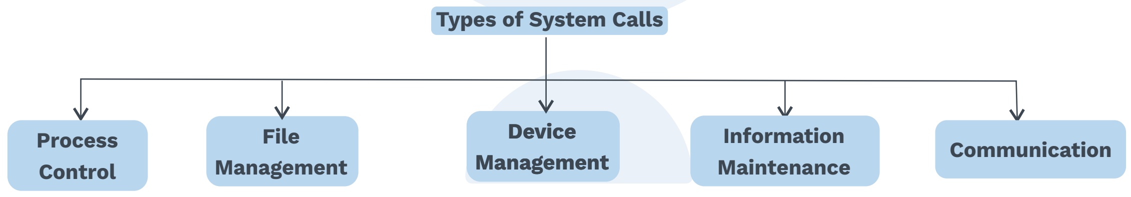


Fig. 1.18 : Types of System Calls

Process Control:

Process Control			
1)	end, abort	**2)**	load, execute
3)	create process, terminate process	**4)**	wait for time
5)	wait event, signal event	**6)**	allocate and free memory

Table 1.1

File management:

File Management	
1)	create file, delete file
2)	open, close
3)	read, write, reposition

Table 1.2

Device management:

Device Management	
1)	request device, release device
2)	get device attributes, set device attributes

Table 1.3

Information maintenance:

Information Maintenance	
1)	get time or date, set time or date
2)	get system data, set system data

Table 1.4

Communication:

Communication	
1)	send messages, receive messages
2)	transfer status information
3)	create, delete communication connections

Table 1.5

System call examples:

1) **wait ( )**

   Some processes are interdependent i.e. execution of one process depends on the other process execution.
   For instance, during fork() if the execution of parent process is dependent on the execution of child process then during the time child process completes its execution, parent process gets suspended and this suspension is done by wait().

2) **exec ( )**

   The purpose of exec() is to execute a .exe file in place of the file which was executing previously.

3) **fork ( )**

   The purpose of fork() is to create a copy of a process itself.
   Execution of fork () at runtime will create a child process i.e a new process.
   Child process code is exact same as the parent one.
   The resources used by child process like cpu registers, PC are same as parent process.

4) **exit ( )**

   The purpose of exit() is to terminate the execution of the ongoing program.
   After using exit() system call, OS can claim again for the resources used earlier by the process.

Rack Your Brain

What would happen if user process can directly access hardware resources without system calls?

1.2 DUAL-MODE IN OPERATING SYSTEM

User mode (Non-Privileged mode) and Kernel mode (Privileged Mode):

- Every CPU operates in the above two modes.
- All operating system services execute in the Kernel mode, with atomic execution i.e. non-preemptively.
- All user applications runs in user mode pre-emptively.
- There is a bit known as mode bit that indicates in which mode processor is currently working.
- Shifting of mode occurs according to the requirement.

e.g.: When a service is requested by a process via OS, mode shift changes from user mode to kernel mode.

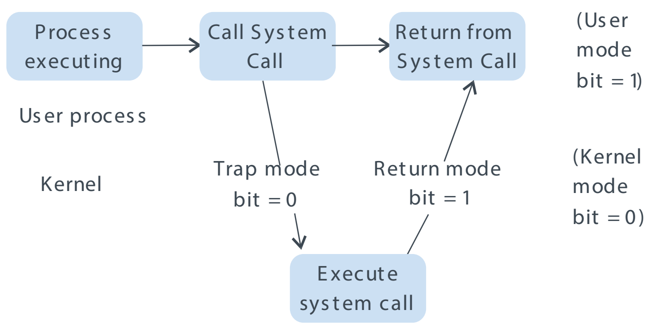


Fig. 1.19 Transition from User to Kernel Mode

- Mode bit is added to the hardware that represents the current mode.
- When mode bit = 0 means representing kernel mode.
- When mode bit = 1 means representing user mode.
- All those instructions that are privileged one must have to run in kernel-mode. They cannot run in user mode otherwise these instructions will be considered as an illegal one.
- Input-Output Control, management of interrupts are the example of privileged instruction.
- When a system starts, the hardware operates in the kernel mode.
- The operating system is then loaded and starts user applications in user mode.
- When an interrupt or system call occurs, it leads to a supervisory call which handles ISR(Interrupt Service Routine), when a supervisory call is generated, ISR changes mode bit in PSW(Program Status Word) from 1 to 0 i.e from user to kernel mode.
- After completion of all the functions in the kernel mode,again supervisory call is generated and mode bit changes from 0 to 1 i.e. kernel to user mode.
- It means OS implements its functionality only when it is in kernel mode.

Rack Your Brain

What would happen if system has only one mode for program execution?

1.3 INTERRUPTS

It is a signal initiated by a software or a hardware when any of the process needs sudden attention.

External interrupt occurs due to hardware.

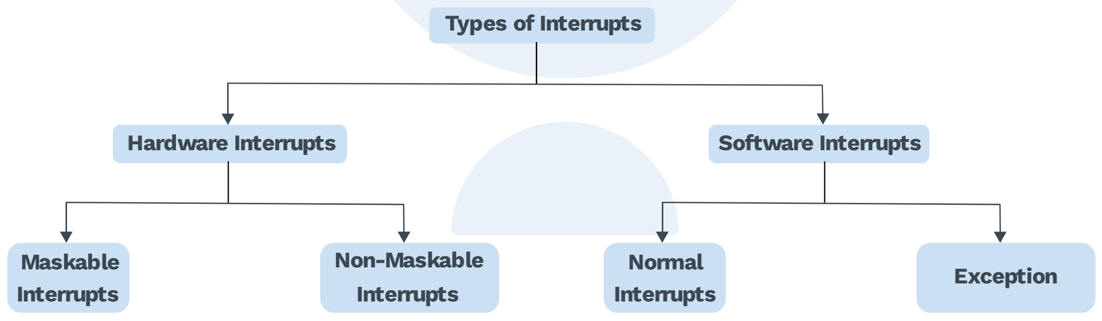
Internal interrupt occurs due to software instructions.

It is also called as software interrupt.
Divide by 0, protection violation are the examples of internal interrupts.

What are the steps to handle a interrupt?
Steps:

- Suppose an interrupt is initiated by a device and by that time process P1 was executing.
- Processor will first complete its execution.
- The address of interrupted instruction is saved to a location that is temporary one.
- After that it will load the PC(program counter) with the address of the 1st instruction of ISR(Interrupt service routine).
- After completion of new request (Interrupted request), processor will start excuting next process P_1 +1.

Types of Interrupts:



1) **Hardware Interrupts:**
   When interrupt occurs due to the external devices.
   i) **Maskable Interrupts:**
      Those interrupts can be disabled if high priority interrupt comes. **eg.** RST 6.5, RST 7.5, RST 5.5 of 8085 microprocessor.
   ii) **Non-Maskable Interrupts:**
      Those interrupts that needs to process immediately. **eg.** Trap of 8085 microprocessor.
2) **Software Interrupts:**
   i) **Normal Interrupts:**
      Those interrupts that occur due to software-instruction.
   ii) **Exception:**
      Some unexpected events occur while a program is executing.
   eg. Trap, Divide by 0 etc.

PRACTICE QUESTIONS

Q1 **Operating systems manage _____.**
a) Only hardware
b) Only user programs
c) Both hardware and user programs
d) Hardware and OS modules only

Sol: **c)**
Apart from managing hardware components, operating systems manage applications programs and other software abstractions like virtual machines.

Q2 **Kernel is _____.**
a) Designed by processor manufacturer
b) Designed by the OS designer
c) Part of processor
d) A central control point of an OS

Sol: **b), d).**
Operating system is a collection of several programs; kernel is the program which controls other programs and services of the OS. Kernel is designed by the OS designer.

Q3 **Can operating systems be portable?**
(Note: Answer 1 for TRUE and 0 for FALSE)

Sol: **1**
Yes, few operating systems are portable which can be directly booted from flash or USB drives. Portable OS is mainly used to recover data on a disk after system crashes.

Q4 **A user is allowed to read/write __________.**
a) In any section of the disk
b) In user specific sections of the disk only
c) In volatile memory only
d) Nowhere in the disk

Sol: **b)**

A user should not be allowed to read/write in any section of the disk because knowingly or unknowingly it can overwrite operating system files or other sensitive data which can cause in crashing the system.

Q5 **Which one of the following statements is correct?**

a) A multi-tasking operating system must be installed on a multiprocessor system to execute multiple processes simultaneously

b) A multi-tasking operating system executes multiple processes simultane–ously on a uniprocessor system

c) A and B both

d) None

Sol: **b)**

A multi-tasking operating system executes multiple processes simultaneously on a uniprocessor system by switching the processes on CPU in time sharing fashion.

Chapter Summary

- **Operating System:** It is an environment, resource allocator, interface between hardware and computer user.

• **Goals of an Operating System**	:	Convenience
		Efficiency
• **Function of Operating System**	:	Process Management
		Memory Management
		I/O Device Management
		File Management
• **Types of Operating System**	:	Batch Operating System
		Multiprogramming Operating System
		Multi-tasking Operating System
		Real-Time Operating System
		Multiprocessor System
• **History Operating System**	:	1st Generation $\rightarrow$ [1940 – 1950] Punch Cards
		2nd Generation $\rightarrow$ [1950 – 1970] Magnetic Tapes
		3rd Generation $\rightarrow$ [1980 – 1990] Multiprogramming
		4th Generation $\rightarrow$ [2000– Present] Modern OS
• **OS Services**	:	GUI (User Interface)
		Program Execution
		File System Manipulation
		I/O Operations
		Error Detection
		Protection and Security
• **OS Operations**	:	Dual mode
		(i) Kernel mode
		(ii) User mode

S.No.	Kernel Mode	User Mode
1)	Unrestricted and full access to computer hardware	Limited access to system's hardware.
2)	Only core functionality can be allowed to operate	User application are allowed to operate.
3)	System Crashes are fatal and increases the complexity.	System crashes are recoverable.

Table 1.6

- **Booting of an OS:**

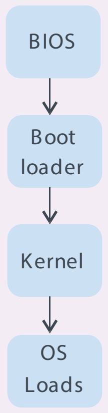


Fig. 1.18 Steps Involved in a Boot Process