26/09/2022

Introduction to Operating System

Topics

* Evaluation
* definition
* Operating system functionalities
* Types of operating systems
* Computer Architecture support to operating systems
* Kernel and user mode
* Introduction to systems calls and function call.

Operating System:-

→ An operating system is a program which manages all the computer hardwares.

→ It provides the base for application program and acts as an intermediary between a user and the computer hardware.

→ The operation system has two objectives such as:

* Firstly, an operating system controls the computer's hardware.
* The second objective is to provide an interactive interface to the user and interpret command so that it can communicate with the hardware.

Advantages of an operating system:-

* Hardware Abstraction.
* Resource Management.

→ Hardware abstraction means hiding the details of different hardware configurations so that each application doesn't have to be tailored for each possible device that might be present on the system.

→ Operating system is also called as a Resource Manager, because operating system allows multiple programs to be in memory and run at the same time.

Generally a computer system is a collection of hardware and software components designed to provide an effective tool for computation.

→ Hardware refers to the electrical, mechanical and electronic parts that make up the computer. examples are CPU, memory, monitor etc.

→ Software is said to be as a tested program along with the documentation. (user manual).

Software is basically two types:

<i> Application software.

<ii> System software.

<i> Application software :-

An Application software is a computer software package that performs a specific task directly for an end user.

→ To run the application software, system software is must needed.

Examples for an application software :-

Google chrome, fire-fox, Whatsapp, Calculator etc.

<ii> System software :-

System software provides the environment to the application software.

Examples for System Software:-

macOS, Linux, Android, and Microsoft Windows etc!.

Need of an operating system:

We known operating system is an interface between user and hardware.

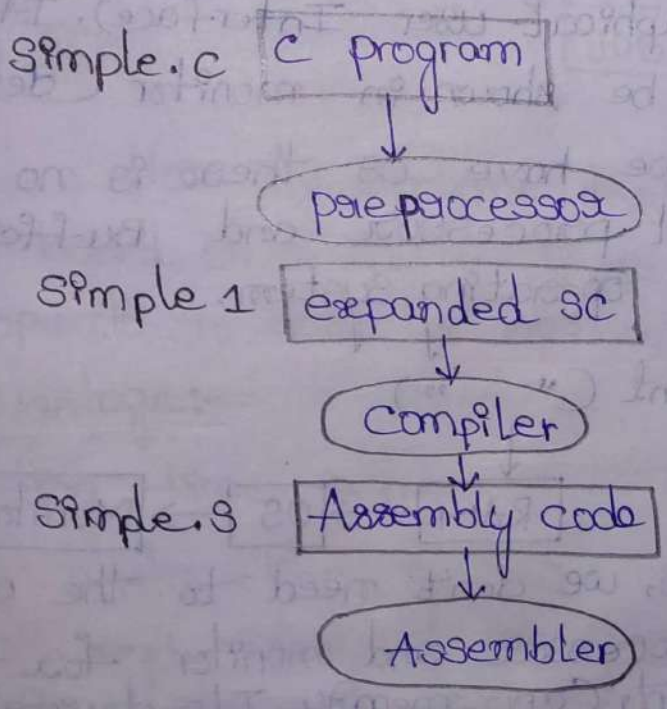
→OS creates user-friendly environment.

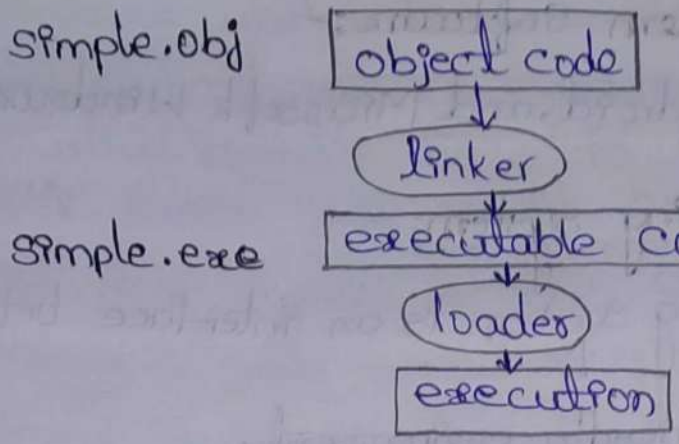
Suppose when working with DOS-OS, if the user wants to delete the program, he has to type the command C:\DEL FILENAME and press the enter, then the program will be deleted. So, the user delete the program very easily with the help of OS.

Suppose, user wants to delete the program without using OS, then he has to write a seperate progam for DEL command and perform the operation. Everytime for doing any operation he has to write a seperate program. So it is very different for the programmer.

So, that OS is used to decrease the burden on the user. It provides user-friendly environment. It is the main need of the operating system.

Execution of C program:-





→ Here linker and loader are system softwares
Linker – which links the files. It generate executable code.

Loader – Executable code is sent to loader which loads into memory and then it is executed.

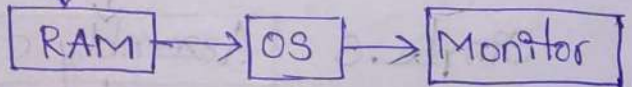
Printf (" "); → program

→ The executable code is load to main memory (RAM). If OS is not present
The loaded file is given to processor it loads the code byte-by-byte.

→ The byte code is given ~~the~~ to the Buffer, by using GUI (Graphical user Interface). It tells how the output will be shown in monitor (desk top).

If we have OS there is no need to understand what processor and Buffer can do. It all manage by operating system.

paint (" ")



If we have OS, we don't need to the write program for processor and monitor for every hardware components (cpu, memory, I/o devices).

29/8/22

Goals of the OS:-

<1> Maximum utilization of CPU

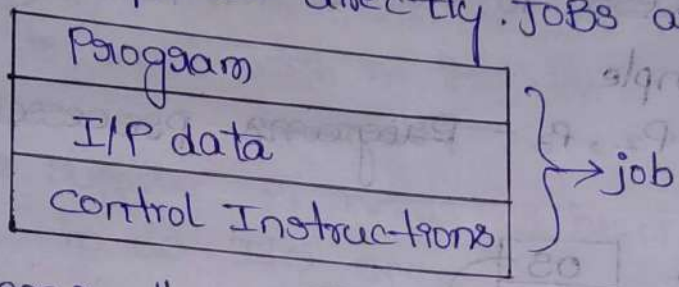
<2> Decrease process starvation (Waiting time for a process).

Types of OS:

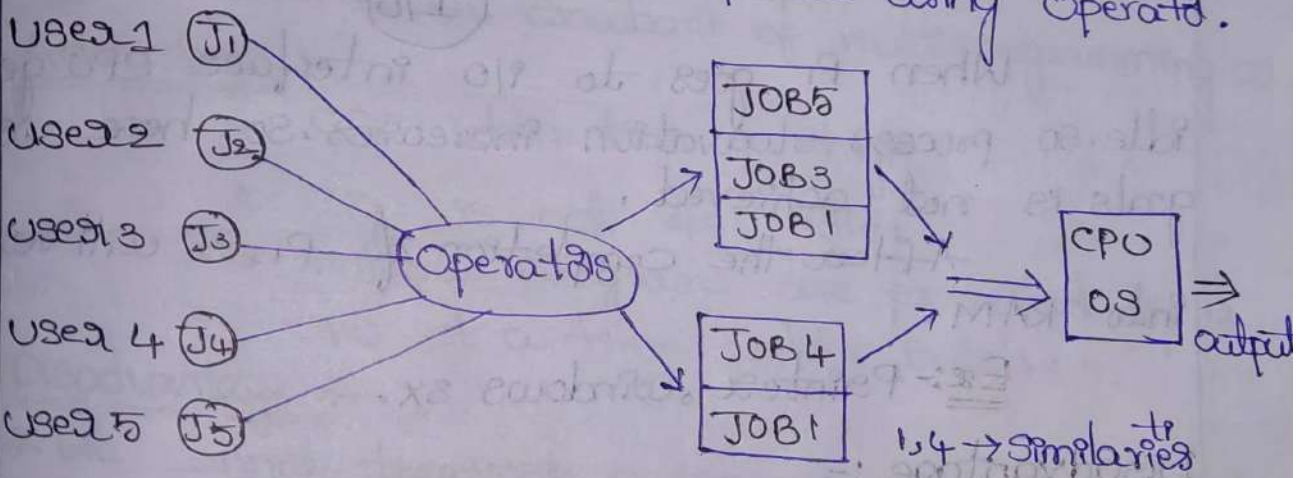
(1) Batch OS :-

These Batch have JOBs.

The users who using a batch os donot interact with the computer directly. JOBs are in punch cards.



User prepare the JOBS in the form of punch cards and submit the JOB to computer using Operator.



Here the operator divide the jobs into batch based on similarity. It is the responsibility of the operator to sort the jobs with similar needs.

Disadvantage:-

* Waiting time is more.
* Cpu utilization is low.
  Cpu kept ideal if job went to o/p devices
* Both goals are not achieved.

Advantages:-

(1) Dividing similar jobs into a batch.

(2) Batch OS is used in 1st generation computers. Through punch cards only the input is given.

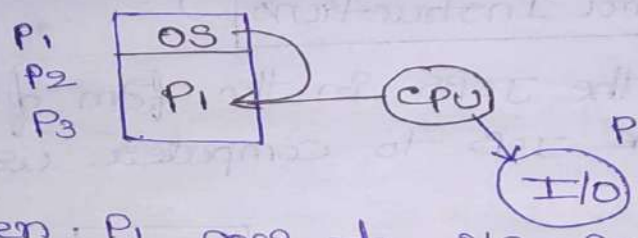
(2) Single Processing OS:-

A single processor operating system contains only one processor.

At a time this processor will allow only one process to reside in RAM (main memory).

For example

P1, P2, P3 — ~~Programs~~ Processes



When P1 goes to I/O interface CPU get idle. So process starvation increases. So here OS goals is not achieved.

After the completion of P1, P2 will reside into RAM.

Ex:- Printer, windows 3x.

Disadvantage :-

By this OS goals are not achieved.

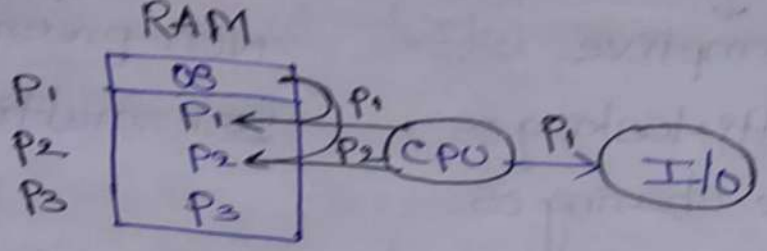
(3) Multi Programming OS:-

An operating system that executes more than one ~~program~~ process using a single processor.

(or).

At a time it can allow multiple ~~program~~ Processes in RAM.

Let us take 3 processes
P_1, P_2 and P_3



After scheduling P_1 to CPU, if P_1 wants to perform I/O. Then CPU will get idle at this stage. At this time OS schedule P_2 to CPU.

* If P_1 and P_2 both need to do I/O operations, then P_2 has to wait. With in this time CPU will execute P_3.
* Let us suppose P_1 needs 1 hour time and P_2 needs 1 minute to do I/O operations. even though P_2 needs less time it has to wait until the P1 is completed.

This is the drawback of multiprogramming OS.

Advantage :-

* CPU utilization is high.
* CPU is not kept idle.
* multiple programs are executed by a CPU at a time unlike Batch OS.

Disadvantage :-

* We cannot terminate a process from CPU forcefully.
* It has to terminate itself.

Degree of multiprogramming :-

It is used to increase the no. of processes in main memory such that CPU cannot get into idle stage.

Multi Programming

- Preemptive
  - ex:- Multi-tasking OS
  - Time sharing OS
- Non-preemptive
  - ex:- multiprogramming OS.

Preemptive :- We can force-fully terminate the process.

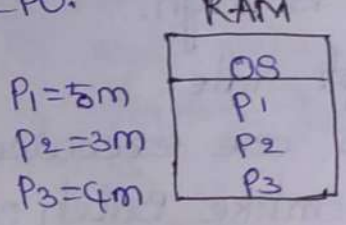
Non-Preemptive :- We can't terminate the process forcefully in the CPU.

→ Preemptive force-fully terminate the process by using time quantum.

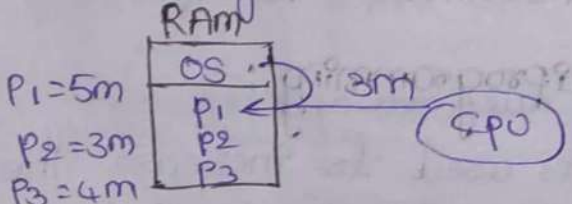
***Multi-tasking OS :-**

Logical extension of multi-programming OS is called multi-tasking OS.

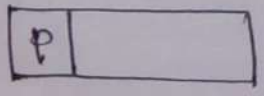
For example if we have P_1, P_2 and P_3 processes and CPU.

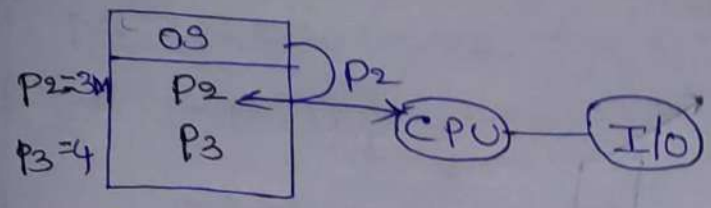


P_1 takes 5 min to complete the process, P_2 takes 3 min and P_3 takes 4 min to complete the process. But OS give 3 min to complete the every process.



P_1 doesn't complete it's task within 3 min. So, OS force-fully removes the P_1 from CPU. and then it will wait and goes to ready queue.





P2 complete it's task in 3 min. Then OS will scheduling the P3 process to the CPU, P3 will also doesn't complete it's task with in 3 min time. Then it will goes to ready queue.

P1	P3

Again OS will scheduling the P1 Process from ready queue to the CPU. Already P1 ~~60%~~ completed 60% task in 3 min time, so remaining 40% will be completed in 2 min time.

After completion of P1 process, again OS will scheduling the P3 process from ready queue to the CPU. This time P3 has remaining 2 min to complete it's task. So, P3 is also completes it's task within time quantum.

∴ P3 is completely terminated.

* This process will continues until all the processes are completed.

Advantage:- *equal time is given to each process.

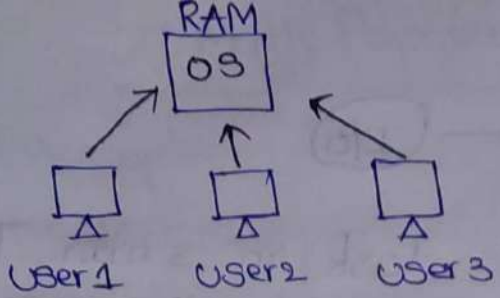
* Goals of the OS will be achieved.

* Context switching.

context switching means, it is used to allow multiple processes to the RAM, but it will allot only one process to the CPU.

<4>. Multi-User OS:-

Multiple users are going to utilize the single Operating system.

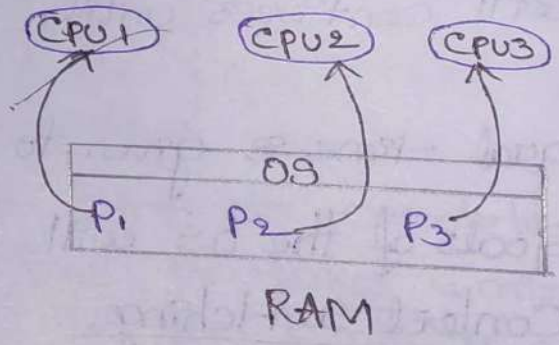


Ex:- Linux, Unix, mac os x, ubuntu, windows etc., are the examples of multi user OS.

(5). Multi-Processor OS:-

Here multiple CPU are there but all these have only single operating system.

* Mutiple processors / multiple CPU's are running parallely at the same time.
* Let us take CPU1, CPU2, CPU3 and processes P_1, P_2, P_3. If P_1, P_2, P_3 processes are in main memory then let P_1 is allocated to CPU1, P_2 is allocated to CPU2, P_3 is allocated to CPU3 } at the same time.



Advantage:-

* multiprocessor is used to decrease process starvation.
* If one CPU will fail we can use another CPU.

ex:- windows NT, Unix are examples of multi-processor OS.

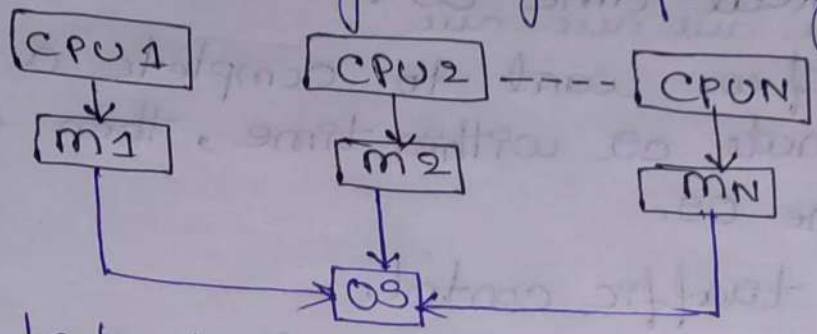
Multi-processor OS is divided into 2 types.

<1> Tightly coupled ex:- shared OS.

<2> Loosely coupled ex:- Distributed OS.

* shared OS :-

In this every processor have it's own memory. It has only single operating system.



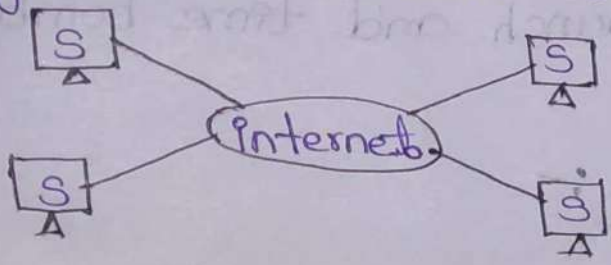
→ It has high data rate.

Dual core - 2 processor for single system.

Quad core - 4 processor for single system.

* Distributed OS :- / Network OS

→ Each system has it's own OS, hardware and CPU.



→ These is distributed OS.

→ These systems are connected to through internet.

→ Through with the help of internet different systems are connected.

(6). Real time OS:-

It is an advanced OS. It works on time interval.

→ It has two types. They are

(i) Soft Real time OS.

(ii) Hard Real time OS.

(i) Soft Real time OS :-

If we want to complete a task in approximate, or within time, then we use Soft Real time OS.

Ex:- air traffic control.

→ Here time will fluctuate with in fraction of seconds.

(ii) Hard Real time OS :-

If we want to complete a task at exact time then we use Hard Real time OS.

Ex:- Missile launch and time bomb etc.,

13/10/2022

*Function call :-

In order to execute the program we can call the function.

*System call :-

System call acts as an interface between a process and an operating system (Kernal).

→ For communicating with OS, we can't directly interact with OS, we use a language that is called API. [Application Prograprogramming Interface].

→ API is used to directly communicate with the OS.

→ For getting available resources from OS we are making use of system calls.

→ system call Provides the services of OS to the process via API.

→ Whenever the process is being executed and if it requires any resource, the Process will create a system call (called interrupt) and sends to the Kernal. (OS).

→ By execution of this system calls the Kernal will give the resource to the Process.

Different types of system calls

1. Process
2. Device
3. Communication
4. File
5. Information.

(1) Process :- These system calls deals with processes such as process Creation, Process termination etc.

(2) Device :- These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

(3) Communication :-

These system calls are useful for inter process communication. They also deal with creating and deleting a communication connection.

<4> File :-

These system calls are responsible for the file manipulation such as creating file, reading a file, writing into a file etc.

<5> Information :-

These system calls handle information and it transfer between the operating system and the users programming.

Types of system calls	Linux.
Process	fork ( ) exit ( ) wait ( ),
File	open ( ) read ( ) write ( ) close ( )
device	ioctl ( ) read ( ) write ( )
information	get pid ( ) alarm ( ) sleep ( )
communication	Pipe ( ) shmget ( ) mmap ( )

Modes:

There are two types of modes.

(1) User mode

(2) Kernel mode.

* A processor in a computer running windows has two different modes: User mode and Kernel mode.
* The Processor switches between the two modes depending on what type of code is running on the processor.
* Applications run in user mode and core applications OS components run in Kernel mode.

User mode :-

(1) When you start a user-mode application, windows creates a process for the application.

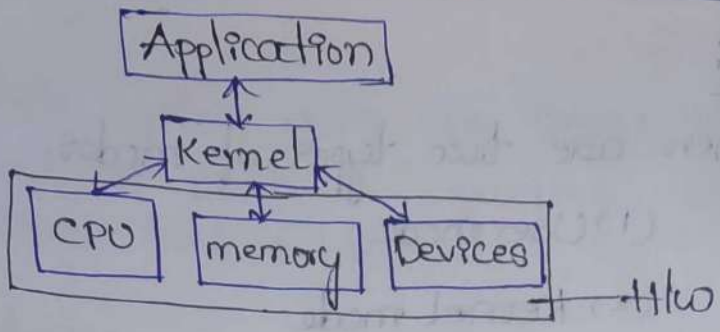
(2) The process provides the application with a private handle table.

(3) It can't access one application to another.

(4) Each application runs in isolation and if an application crashes, the crash is limited to that one application.

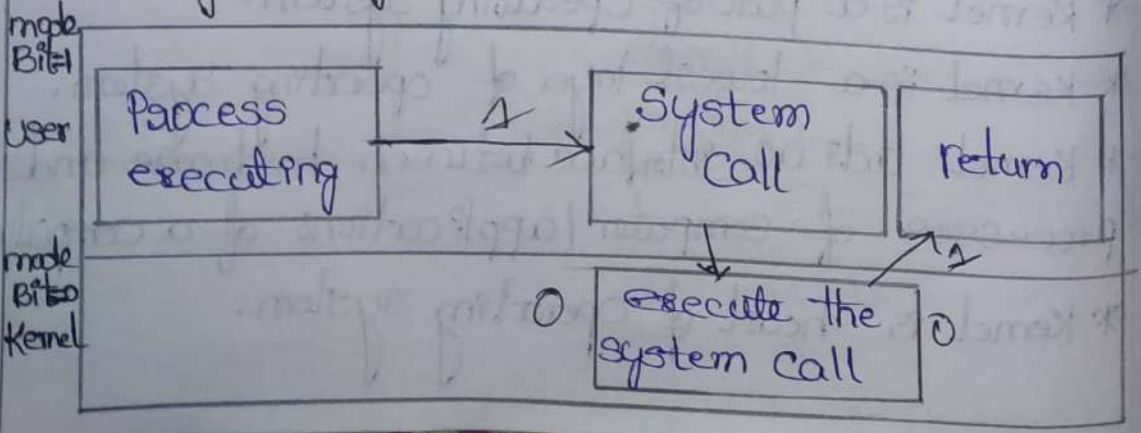
Kernel Mode :-

* Kernel is a part of operating system.
* Kernel is a lowest layer of operating system.
* Kernel acts as interface between hardware and processes of computer / applications of a computer.
* Kernel is heart of operating system.



* Kernel will acts as a interface between soft applications and the hardware.
* It is mediater between user Application and hardware.
* Kernel loads first in memory after OS has been loaded and remains in memory until OS is shutdown.
* Once we power on the system, the booting process will takes the place. i.e, the OS will be loaded and immediately Kernel will load first in memory and will remains in the memory itself until the system gets shutdown.
* It is a system software.
* It allocates resources for the particular task in the Background.
* The operations we performed internally is done by kernel.

Working of system call :-

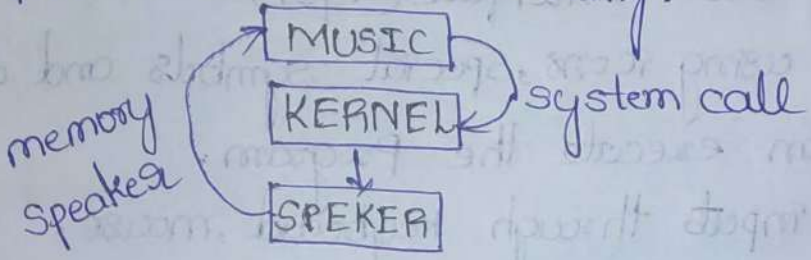


* function call will execute in USER mode.
* system call will execute in KERNEL mode.
* A system call is an interface between USER and KERNEL to avail the resources from OS.

→ Process executing will be done in usermode, getting resources for process executing will be done in Kernel mode.

for example

If we want to play music, initially it will be in usermode and then for audible we will need sound that is speaker (Hardware device). for this device we ask Kernel (Kernel mode) after getting resource we get backed to usermode. This is the process done internally.



→ for switching from one mode to another mode we can make use of TRAP() instruction in system call.

→ Mode bit = 1 is called user mode.
Mode bit = 0 is called Kernel mode.

→ we will consider Kernel as a software interrupts because it is associated with OS.

→ Kernel is a system software.

14/10/2022

Functions of OS:

① Process Management :-

Process :- A process is called when a program under a execution.

→ We execute ourfile in two different ways.

(i) Command Line Interface (CLI)

By using commands we run our programs we get permission for displaying output on monitor. I/o through keyboard only.

ex:- Terminal for Linux.

(ii) Graphical user Interface (GUI)

By using icons, special symbols and clickable items we can execute the program.

→ We give inputs through keyboard, mouse.

→ GUI is a user-friendly because it doesn't need to run commands and remember those commands.

→ CPU peaform actions only on primary memory if data stored in ~~main~~ secondary memory, it should brought to main memory.

(a) secondary memory → mainmemory

As CPU perform opeations on mm it should be within mm, even though data is stored in sm. For retrieving data from sm to mm OS take this responsibility.

sm - Permanent → doesnot erase data even after shut down. Non volatile,

MM - temporary → data get erased and volatile.

(b) Creation :-

we create a file and write into the file.

ex:- hello.c

→ It stored in secondary memory and get backed to main memory. we have to run given hello.c for this we have to ask permissions to OS.

(c) Suspend :-

If we didn't get permission, until that time system would suspend ~~until~~ then CPU get idle.

(d) Resuming :-

After suspension, if we got permission then we can resume our Process.

(e) Synchronization :-

for sharing memory OS make use of scheduling algorithms.

ex:- FCFS

SJF - Shortest Jump first

RR - Round Robing.

(2) Memory Management :-

(a) Primary memory / main memory

→ MM has limited memory space

→ we want to store more than the space which is in main memory, it can be stored in virtual memory.

→ Exceeds of main memory may crash the OS.

→ OS take the responsibility to share in main memory.
→ OS is going to allot memory space by 2 techniques.

1. Continuous — [memory fixed partition.
   memory variable partition.
2. Non-Continuous — [paging
   segmentation.

(ii) Secondary memory :-

The following techniques are the techniques used to allocate the space in secondary memory.

1. Disk management / scheduling.
2. Indexed allocation.
3. Linked List allocation.

(3) File management :-

OS perform read, open, write operations on files.

(4) Device Management :-

OS take responsibilities of alloting device drives based on request and requirements.

(5) Input/Output Managements :-

Based on requirements OS will give input and output.

(6) Protection and security :-

OS provides Authentication and Authorization (permissions) to the file through the help of

debuggers.

debugger - It is used to identify errors and bugs and make necessary actions.

→ We get segmentation errors when we run out of the memory.

→ If we didn't resolve the errors, OS can may crash and then os take responsibility of the errors.

Operations:-

1. Dual mode — User mode 1
   Kernel mode 0
2. Time

→ We can know whether the hardware components are under in which mode using mode bits.

→ If os didn't give resources for much time we make use of preemptive, if preemptive also didn't work system may crash.

→ To overcome these we use Time Operation.

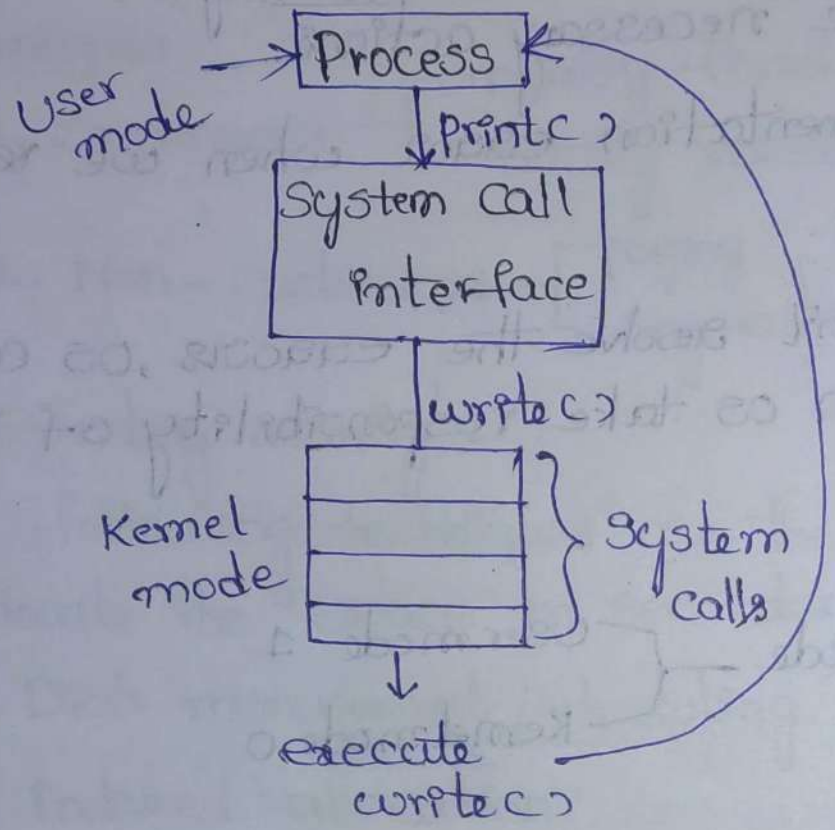
→ As a users we can't directly communicate with kernel. We can communicate through the API.

→ API provides set of prede-fined functions.
For a single task we different system calls, they are

1. Creating
2. Open
3. Checking
4. Error
5. display
6. Close
7. terminate

these system calls are hidden by API and the library which we use in our program.
ex:- stdio.h.

for windows – windows, POSIX – Linux, Java API – JVM, these are the different API's.



* Based on index numbers, system calls will arrange.