Methodologies for implementation of o/s Service
System calls :-

For performing any operation a user must have to Request for a service call which is also known as the System Call.

Or we can say

→ Programming interface to the services provided by the O.S.

→ They are typically written in a High-level language (C or C++)

→ There are two modes in the operation of System which is user mode or System mode

In User mode → all user processes are executed.

In System mode → All privileged operations are executed

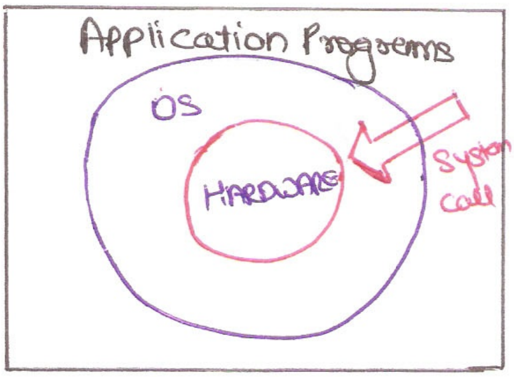
The user programs and kernal functions are being executed in their respective space allotted in the main memory partitions.

User mode programs need to excute some privileged operations, which are not permitted in user mode functions, user mode programs must use an interface, which forms the only permitted interface between user mode and kernal mode. This interface is called System Calls. So System calls is an interface b/w the user programs and the o/s

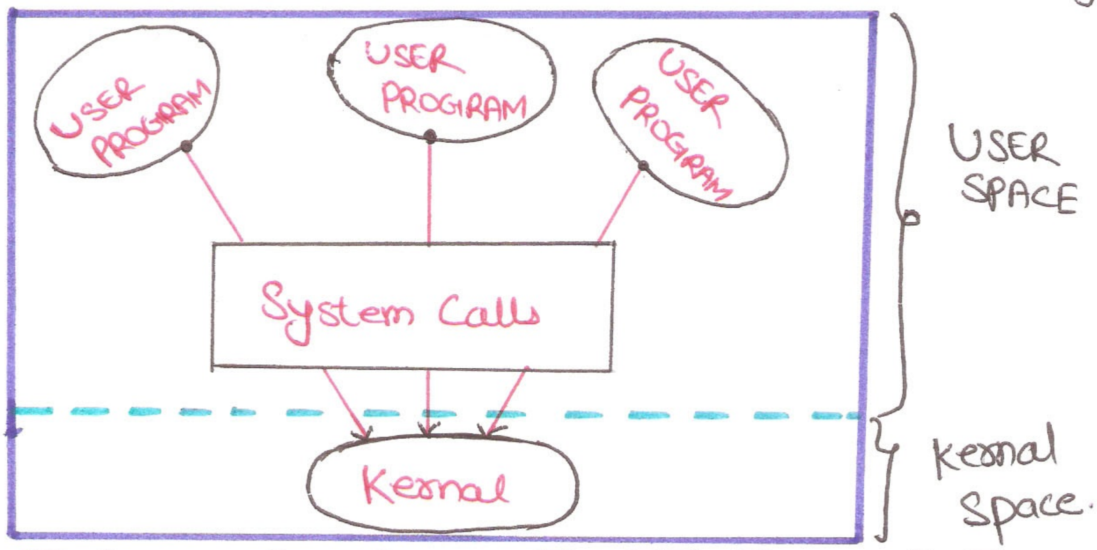
System calls expose all kernel functionalities that user-mode programs require.

Basically the System call is an instruction that request the OS to perform the desired Operation that needs hardware access or other privileged Operations.

* System call generates an interrupt that Causes the OS to gain Control of the CPU. The OS then finds out the type of System Call and the Corresponding Interrupt Handler Routine is executed to perform the operation.



System Calls are inherently used for security reasons. Due to the use of System Calls, a user program is not able to enter into OS or any Other's User's Region. Similarly, I/O devices are also safe



Kernal space + User space in Main Memory from any misuse by the user. Thus, through the use of System calls, Kernel, other user programs, and I/O Devices are safe and secure from malicious user programs

Making a System Call

Now System Calls are directly available and used in high level languages like C & C++. So it has become easy to use System calls in programs.

For a programmer, System Calls are same as calling a procedure or function.

The difference between a System call and a normal function call is that a System call enters a Kernal but a normal function call does not.

Executing the System Call:-

There is a Sequence of steps to execute a System call. for this, there is the need to pass various parameters of System Call to the OS. For passing these parameters to the OS, three methods are used as follow:-

1) Register Method, wherein the parameters are stored in registers of the CPU.

2) If parameters are not in number, Compared to the size of registers, a block of memory is used and the address of that block is stored in Register

3) Stack Method, where in parameters are pushed onto the Stack and popped off by the OS.

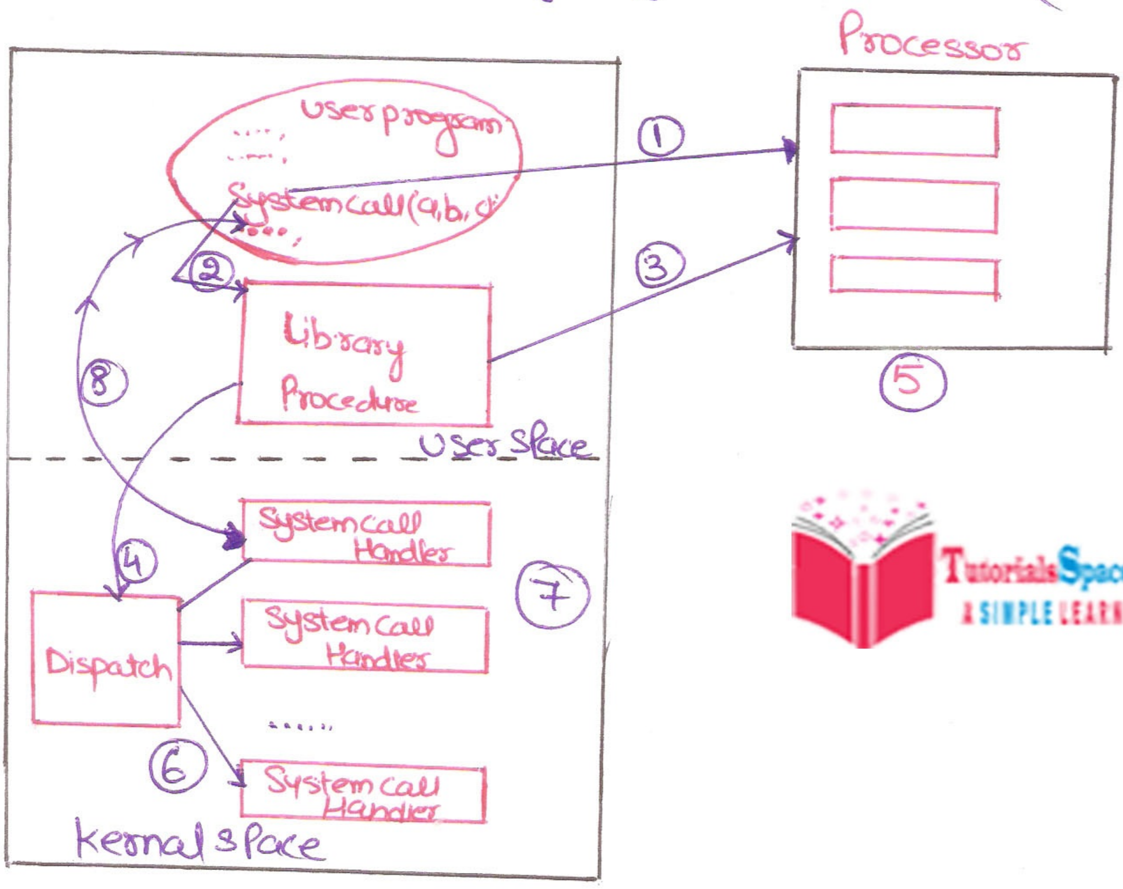
Sequence in which System Call is executed

1) In the User program when the System Call is executed, first of all, its parameters are pushed onto the stack and later on saved in processor Registers.

2) The Corresponding library procedure for the System call is executed.

3) There is a particular Code for every System Call by which the Kernal identifies which System call function or handler needs to be executed. Therefore, library procedure places the System Call number in the processor Register.

4) Then the library procedure traps to the kernal by executing interrupt instruction. With this interrupt execution, the user mode switches to kernal mode by loading 'Program Status Word' (PSW) register to 0.



5→ The hardware saves the Current Contents of CPU registers, so that after executing the System Call, the execution of the rest of the program Can be resumed.

6→ The Kernal identifies the System Call by examining its number and dispatches the Control to the Corresponding System Call handler

7→ The System Call handler executes.

8→ On Completion of System Call handler, the Control is returned to the user program and it resumes its execution.