REPORT

How syscall works?

High Level Overview:

- System call is a way in which a program requests a service from the OS.
- When the user program calls a system call, then the Operating system switches to Kernel mode from User mode. Once after handling the system call, then OS switches to user mode from kernel mode.

Low-Level Overview:
The user program that wants to make a syscall, calls the relevant library routine (like fork()) . The library version of the system call just keeps number corresponding to the syscall in the **%eax** register and raise a trap (or interrupt) with a code corresponding to syscall (in Usys.S) . Any other arguments passed to this library routine are passed to stack.

As a result of trap, the OS switches from user mode to kernel mode. Based on the number in the **%eax** register, the corresponding syscall is identified (using the syscalls table) and the syscall is handled (in syscall.c). While executing the syscall, the arguments passed (to the library routine) are retrieved from the stack (discussed below) and used. The return value is stored in the **%eax** register and passed back to the user, and the OS comes back to user mode.

How parameters are passed to syscall?

In the user.h, we declare the library routine of the corresponding syscall with the parameters and return values.(for example: int mydate(struct rtcdate*)).
While using the syscall in any of the user defined programs, we pass the arguments to the library routine just as ordinary functions.
The syscall just takes the void parameter. (for example: int sys_mydate(void)). Here,we can retrieve the arguments (in sysproc.c), passed using the following inbuilt XV6 functions:

- **argint** for retrieving argument as an integer
- **argptr** for retrieving arguments as a pointer
- **argstr** for retrieving arguments as a string
- **argfd** for retrieving arguments as file descriptor

Overview of syscall working in mydate.c:

In mydate.c, when we call the mydate(r), the number corresponding to syscall(here, sys_mydate) is passed to %eax register, r is stored in the stack. A trap corrseponding to syscall is raised. Then OS is switched to kernel mode.
Now, the syscall is handled based on number in %eax. Then the argument ‘r’ is fetched from the stack using argptr in the sys_mydate() function. Using the inbuilt function

cmostime(), we load corresponding date-time values into ‘r’. If any error occurs while fetching arguments from stack we return -1 else 0 for error handling.