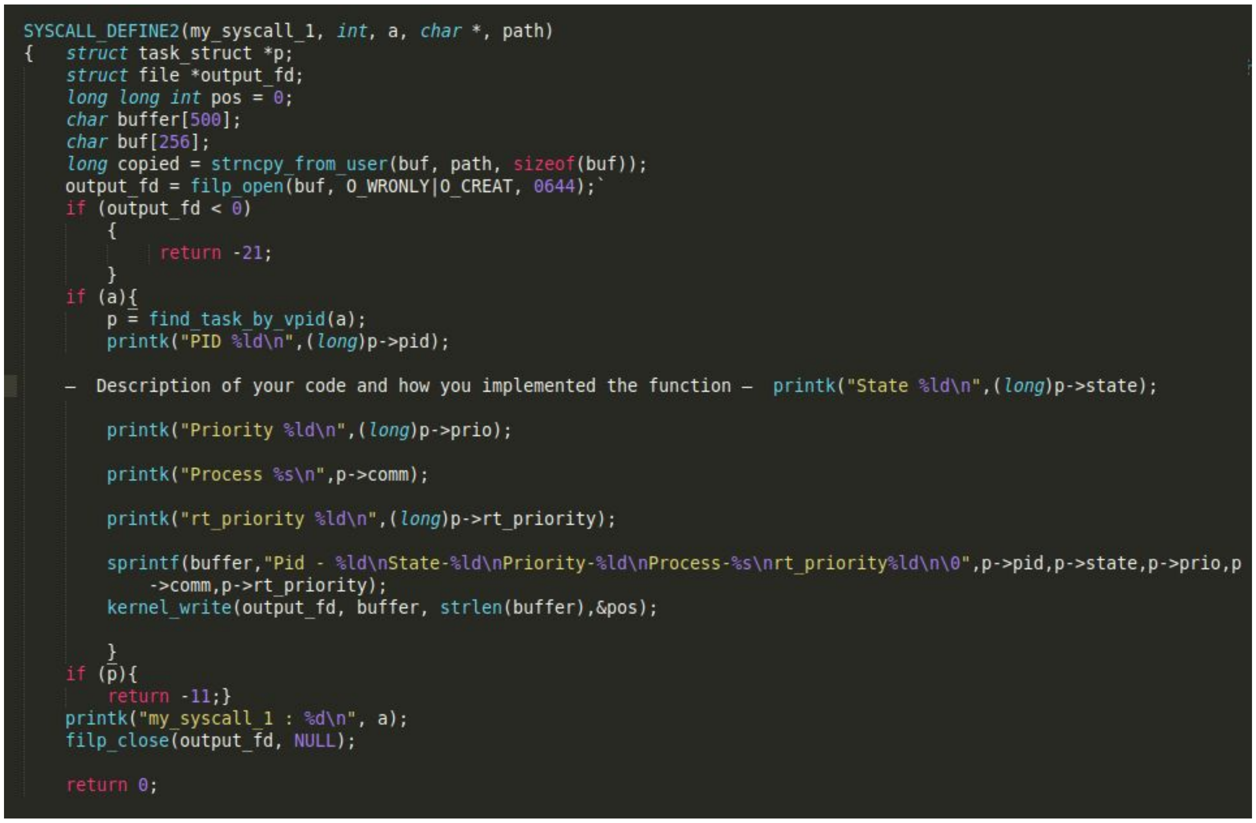
NOTE: While using **make** please use **sudo make** because there may be some permission issue.

Code Implementation:
For test.c file:
In my code I first asked for a PID using the checkPid function then the code asked for the path of the file (**It should be absolute path**). Then using the syscall function of C i called by manually defined system call whose syscall number is **441.**

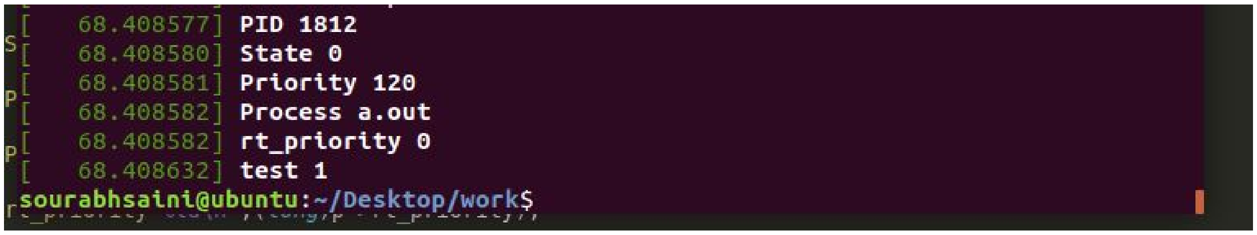
For syscall :



```
SYSCALL_DEFINE2(my_syscall_1, int, a, char *, path)
{   struct task_struct *p;
    struct file *output_fd;
    long long int pos = 0;
    char buffer[500];
    char buf[256];
    long copied = strncpy_from_user(buf, path, sizeof(buf));
    output_fd = filp_open(buf, O_WRONLY|O_CREAT, 0644);`
    if (output_fd < 0)
        {
            return -21;
        }
    if (a){
        p = find_task_by_vpid(a);
        printk("PID %ld\n",(long)p->pid);

    –  Description of your code and how you implemented the function –  printk("State %ld\n",(long)p->state);

        printk("Priority %ld\n",(long)p->prio);

        printk("Process %s\n",p->comm);

        printk("rt_priority %ld\n",(long)p->rt_priority);

        sprintf(buffer,"Pid - %ld\nState-%ld\nPriority-%ld\nProcess-%s\nrt_priority%ld\n\0",p->pid,p->state,p->prio,p
            ->comm,p->rt_priority);
        kernel_write(output_fd, buffer, strlen(buffer),&pos);

        }
    if (p){
        return -11;}
    printk("my_syscall_1 : %d\n", a);
    filp_close(output_fd, NULL);

    return 0;
```

In the above code I first defined all the variables that I needed for the program then copied the user space file path to kernel space using **strncpy_from_user.** Then using filp_open i opened the file i found out the tast_struct corresponding to the PID given and

then using printk i printed task_struct details to kernel log then using kernel_write i wrote these details to file.
Kernel Log:



```
[   68.408577] PID 1812
[   68.408580] State 0
[   68.408581] Priority 120
[   68.408582] Process a.out
[   68.408582] rt_priority 0
[   68.408632] test 1
sourabhsaini@ubuntu:~/Desktop/work$
```

Error Handling:
I used the checkPath function to check that file exit or not at the time of user input or given path is correct or not.Also for PID i handled the case when PID is negative.

User Input:
Program asks the user for input 2 times, first for PID and then for path.
Note: You have to take the absolute path.

Expected Output:
When we compile the code it shows nothing to the terminal but when we do **sudo dmesg** it shows the details of task_struct. And it also writes these details in file.