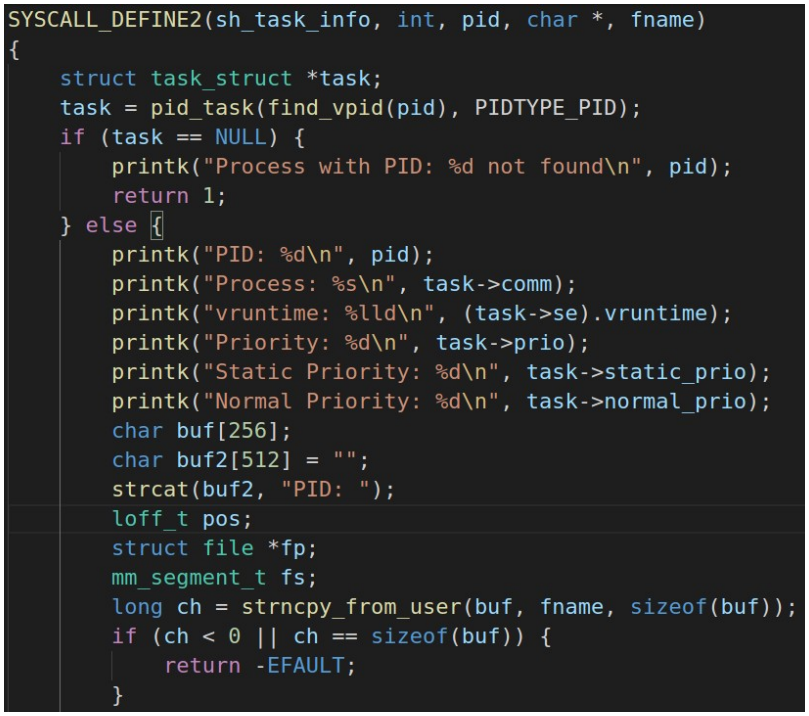
CSE 231: OS Assignment 2

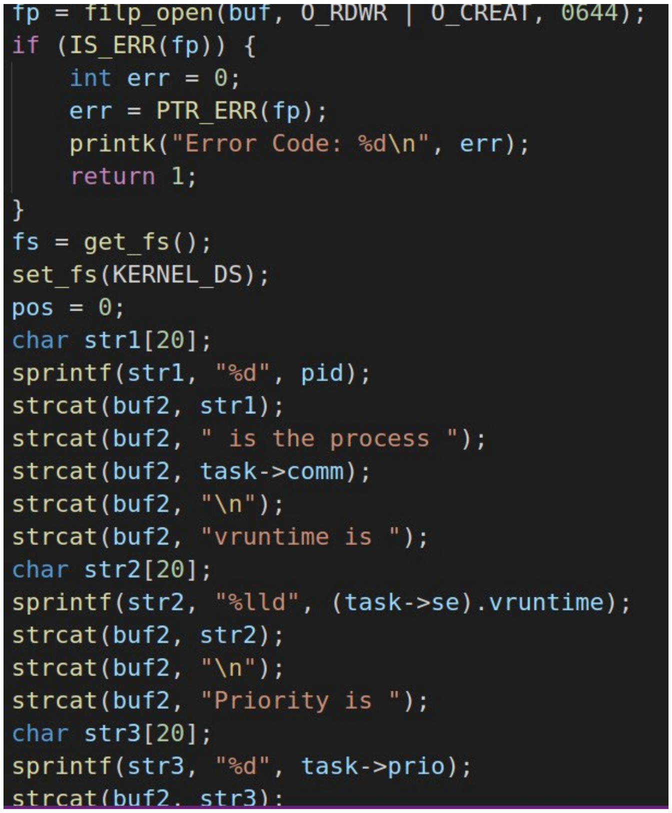
Writing Your Own System Call

1 Description and implementation of the code

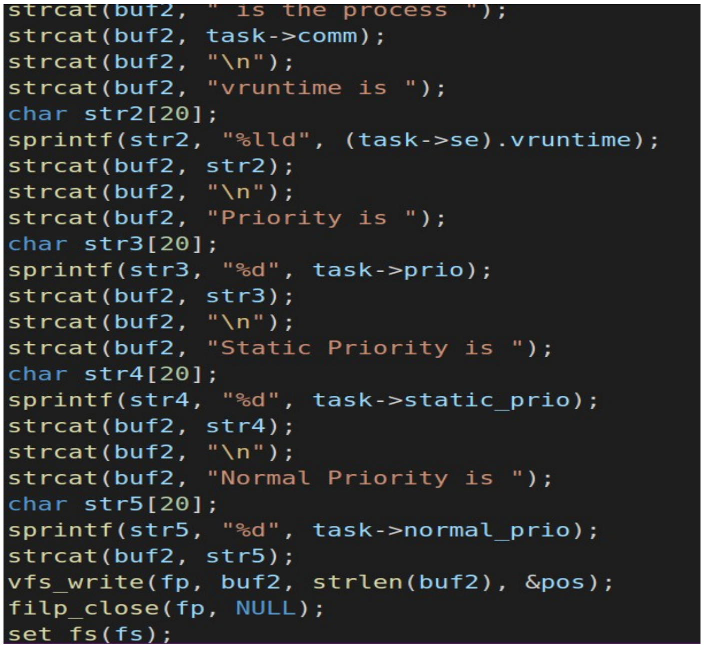
- SYSCALL_DEFINE2 was used as 2 input parameters were passed
- The two parameters were the pid of the required function and the name (along with the required path) of the required output file name.
- pid_task() was used to find the task_struct of the corresponding pid.
- Using the found task_struct, the required fields were printed.
- The printed fields include:
  - pid
  - task name
  - vruntime of the task
  - priority
  - normal priority
  - static priority
- A file was created using the path taken as input.
- filp_open() function was used to do the same.
- If the file with the current name is not present, a new file would be created, else the required data would be written over in the file.
- To do this, all the data would be first converted to string (character array)
- If the process is successful, the syscall would return 0 else 1.



```
SYSCALL_DEFINE2(sh_task_info, int, pid, char *, fname)
{
    struct task_struct *task;
    task = pid_task(find_vpid(pid), PIDTYPE_PID);
    if (task == NULL) {
        printk("Process with PID: %d not found\n", pid);
        return 1;
    } else {
        printk("PID: %d\n", pid);
        printk("Process: %s\n", task->comm);
        printk("vruntime: %lld\n", (task->se).vruntime);
        printk("Priority: %d\n", task->prio);
        printk("Static Priority: %d\n", task->static_prio);
        printk("Normal Priority: %d\n", task->normal_prio);
        char buf[256];
        char buf2[512] = "";
        strcat(buf2, "PID: ");
        loff_t pos;
        struct file *fp;
        mm_segment_t fs;
        long ch = strncpy_from_user(buf, fname, sizeof(buf));
        if (ch < 0 || ch == sizeof(buf)) {
            return -EFAULT;
        }
```



```
fp = filp_open(buf, O_RDWR | O_CREAT, 0644);
if (IS_ERR(fp)) {
    int err = 0;
    err = PTR_ERR(fp);
    printk("Error Code: %d\n", err);
    return 1;
}
fs = get_fs();
set_fs(KERNEL_DS);
pos = 0;
char str1[20];
sprintf(str1, "%d", pid);
strcat(buf2, str1);
strcat(buf2, " is the process ");
strcat(buf2, task->comm);
strcat(buf2, "\n");
strcat(buf2, "vruntime is ");
char str2[20];
sprintf(str2, "%lld", (task->se).vruntime);
strcat(buf2, str2);
strcat(buf2, "\n");
strcat(buf2, "Priority is ");
char str3[20];
sprintf(str3, "%d", task->prio);
strcat(buf2, str3);
```

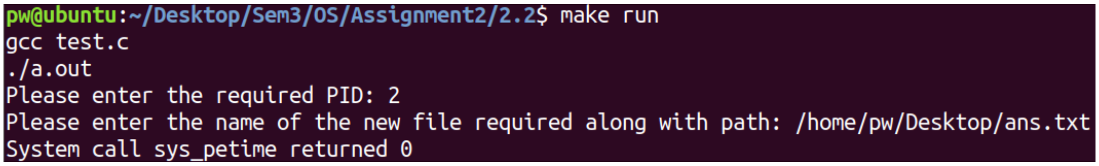


```
strcat(buf2, " is the process ");
strcat(buf2, task->comm);
strcat(buf2, "\n");
strcat(buf2, "vruntime is ");
char str2[20];
sprintf(str2, "%lld", (task->se).vruntime);
strcat(buf2, str2);
strcat(buf2, "\n");
strcat(buf2, "Priority is ");
char str3[20];
sprintf(str3, "%d", task->prio);
strcat(buf2, str3);
strcat(buf2, "\n");
strcat(buf2, "Static Priority is ");
char str4[20];
sprintf(str4, "%d", task->static_prio);
strcat(buf2, str4);
strcat(buf2, "\n");
strcat(buf2, "Normal Priority is ");
char str5[20];
sprintf(str5, "%d", task->normal_prio);
strcat(buf2, str5);
vfs_write(fp, buf2, strlen(buf2), &pos);
filp_close(fp, NULL);
set_fs(fs);
```

- The above are some snippets of the code written for the system call.

2 Inputs from the user

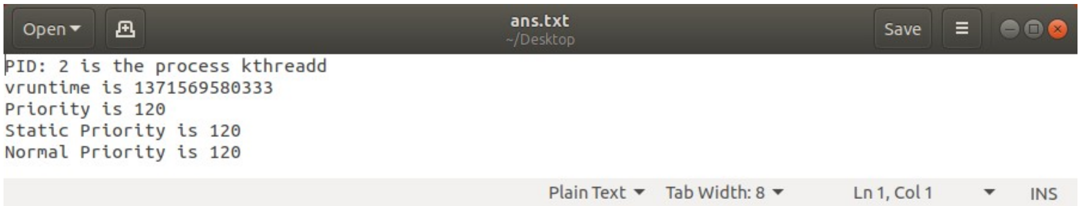
- The user is required to enter 2 inputs:
  - pid of the required process
  - the name (along with the address as to where to store) of the file to be created or written into while executing the system call.



```
pw@ubuntu:~/Desktop/Sem3/OS/Assignment2/2.2$ make run
gcc test.c
./a.out
Please enter the required PID: 2
Please enter the name of the new file required along with path: /home/pw/Desktop/ans.txt
System call sys_petime returned 0
```

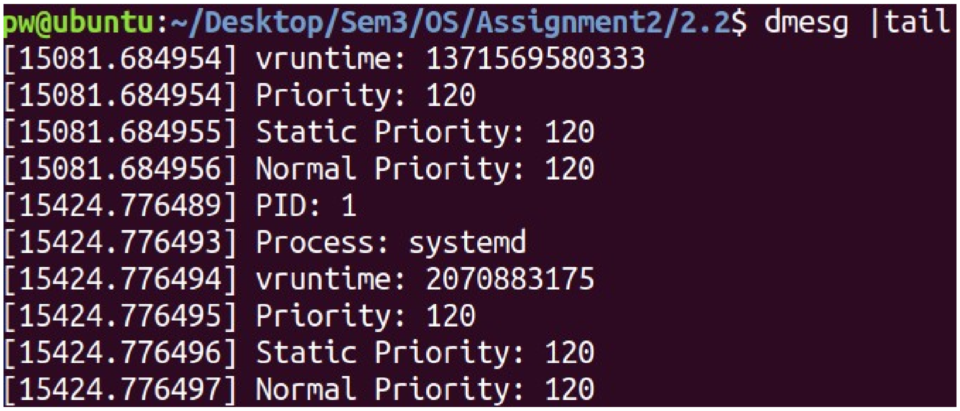
3 Outputs

- If the inputs are correct, the file containing the data is created.



```
PID: 2 is the process kthreadd
vruntime is 1371569580333
Priority is 120
Static Priority is 120
Normal Priority is 120
```

- The data is also printed in the kernel logs and it can be displayed using the command dmesg | tail.



```
pw@ubuntu:~/Desktop/Sem3/OS/Assignment2/2.2$ dmesg |tail
[15081.684954] vruntime: 1371569580333
[15081.684954] Priority: 120
[15081.684955] Static Priority: 120
[15081.684956] Normal Priority: 120
[15424.776489] PID: 1
[15424.776493] Process: systemd
[15424.776494] vruntime: 2070883175
[15424.776495] Priority: 120
[15424.776496] Static Priority: 120
[15424.776497] Normal Priority: 120
```

- In case of an error like invalid pid error message is stored in the kernel log and file is not created.

4 Error Values

- If everything runs fine, the syscall returns 0.
- If the there is an error, 1 is returned.
- Error handling has been done in both the syscall implementation.
- If a wrong(invalid) pid is entered, i.e. a process with that pid does not exists then we get a corresponding message. In such a situation, the file is not created.
- Similarly, other errors like invalid path name, invalid file size (it has been assumed that the size of file including the path can-not exceed 256 characters) etc. also give their corresponding error messages.
- The above error messages can be viewed by typing the command dmesg |tail.

5 Diff File

- Three different diff files are supplied with this code.
- The diff files were created after executing the make clean command
- The diff files by the name of “diff.txt” and “diffruN.txt” were created by using the flags “-r”, “-u” and “-N”.
- The diff file by the name of “diffruaN.txt” was created by using the flags “-r”, “-u”, “-a” and “-N”.