

OS Assignment 2

Question 2

CODE

Description:

The test.c program simply takes user input (for the process ID and file name), calls sh_task_info, performs error handling and presents the result.

sh_task_info.c defines the system call sh_task_info using the SYSCALL_DEFINE family of functions for 2 arguments (via SYSCALL_DEFINE2). The arguments received the PID of the task (type:pid_t) and name of the file to write to (type: string).

```
struct task_struct *task;  
task = pid_task(find_vpid(pid), PIDTYPE_PID);
```

[Accessing task_struct for specified PID]

We access the task_struct using these lines. If it is not found (if it is NULL), -1 will be returned to test.c and an appropriate message is printed using printk and KERN_ERR.

```
long copied = strncpy_from_user(filename, name, sizeof(filename));  
if (copied < 0 || copied == sizeof(filename)){  
    return -EFAULT;  
}
```

[Copying user specified string to kernel space for use]

We use the strncpy_from_user function to access the string that was passed from user space in the kernel space, something that is required for this program. If this is not possible, an error is thrown.

If neither of the errors is encountered, we extract the required details of the process from the fields of the task_struct and print them using printk.

```
printk(KERN_INFO "The common pid found: %d \n", task->pid);  
printk(KERN_INFO "Name: %s \n", task->comm);  
printk(KERN_INFO "Started at: %llu nanoseconds\n", task->start_time);  
printk(KERN_INFO "Priority: %d \n", task->prio);  
printk(KERN_INFO "State: %d \n", task->state);  
printk(KERN_INFO "Exit State : %d \n", task->exit_state);  
printk(KERN_INFO "Parent : %d \n", my_parent->pid);  
  
list_for_each(list, &task->children) {  
    child = list_entry(list, struct task_struct, sibling);  
    printk(KERN_INFO "PIDs of Child Processes : %d \n", child->pid);  
}
```

[Printing the PID, name, start time, priority, state, exit state, parent PID and PID of all children]

If no error is encountered, we write the pertinent fields to the file that has been specified by the user..

```
if (IS_ERR(filp)) {  
    return -1;  
}  
  
else {  
  
    char string [2048];  
    sprintf(string,"Process id: %d\nName: %s\nPriority: %d\nState: %ld\nParent: %d\n",pid,task->comm,task->prio,task->state, my_parent->pid);  
  
    nbytes=strlen(string);  
  
    oldfs = get_fs();  
    set_fs(KERNEL_DS);  
    vfs_write(filp, string, nbytes, 0);  
    set_fs(oldfs);  
    filp_close(filp, NULL);  
}
```

[Writing 5 fields to the user specified text file]

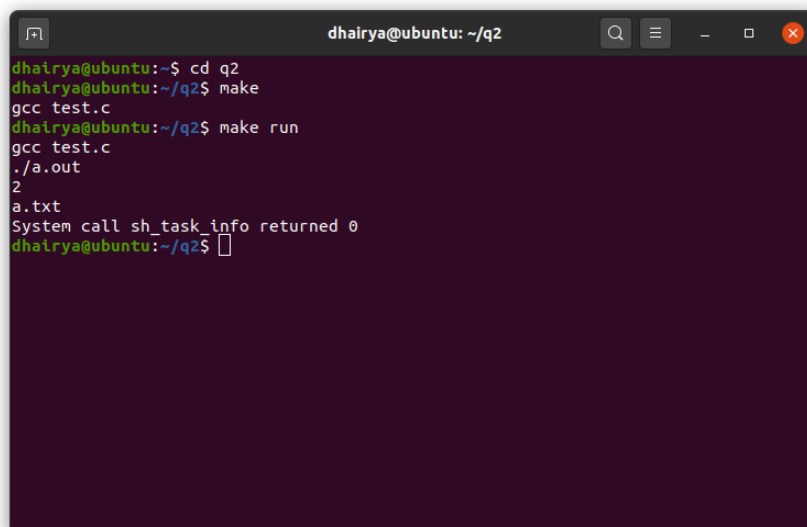
After this, we return 0 and the program has been successfully executed.

Assumptions:

1. Filename provided by the user will not have length greater than 256.

Expected Input:

1. PID
2. Filename

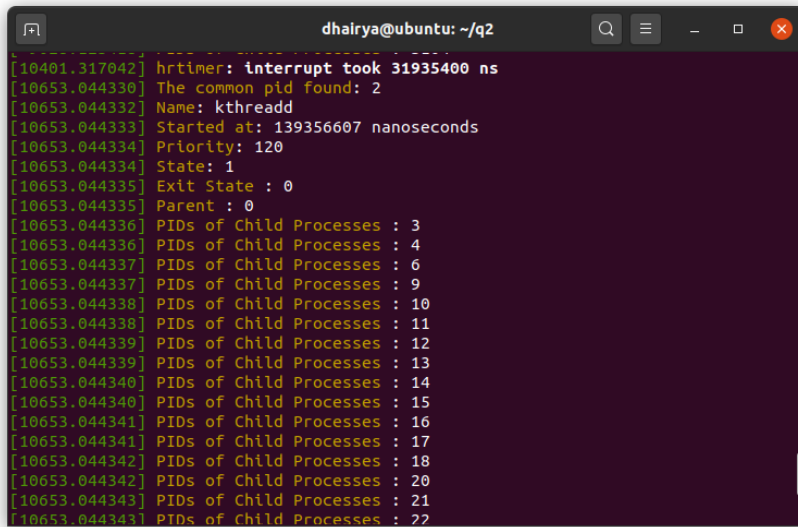


```
dhairya@ubuntu: ~/q2  
dhairya@ubuntu:~$ cd q2  
dhairya@ubuntu:~/q2$ make  
gcc test.c  
dhairya@ubuntu:~/q2$ make run  
gcc test.c  
./a.out  
2  
a.txt  
System call sh_task_info returned 0  
dhairya@ubuntu:~/q2$
```

[Demonstration of sample input]

Output:

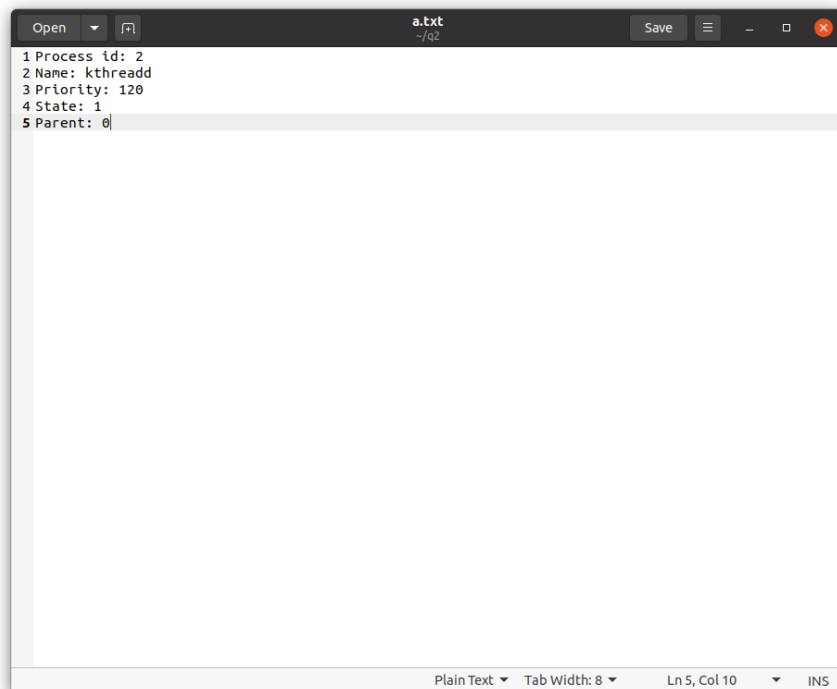
We can see output after typing 'dmesg' command.



```
dhairya@ubuntu: ~/q2
[10401.317042] hrtimer: interrupt took 31935400 ns
[10653.044330] The common pid found: 2
[10653.044332] Name: kthreadd
[10653.044333] Started at: 139356607 nanoseconds
[10653.044334] Priority: 120
[10653.044334] State: 1
[10653.044335] Exit State : 0
[10653.044335] Parent : 0
[10653.044336] PIDs of Child Processes : 3
[10653.044336] PIDs of Child Processes : 4
[10653.044337] PIDs of Child Processes : 6
[10653.044337] PIDs of Child Processes : 9
[10653.044338] PIDs of Child Processes : 10
[10653.044338] PIDs of Child Processes : 11
[10653.044339] PIDs of Child Processes : 12
[10653.044339] PIDs of Child Processes : 13
[10653.044340] PIDs of Child Processes : 14
[10653.044340] PIDs of Child Processes : 15
[10653.044341] PIDs of Child Processes : 16
[10653.044341] PIDs of Child Processes : 17
[10653.044342] PIDs of Child Processes : 18
[10653.044342] PIDs of Child Processes : 20
[10653.044343] PIDs of Child Processes : 21
[10653.044343] PIDs of Child Processes : 22
```

[The earlier specified task_struct fields have been printed]

It is also saved to the specified file (which was also created in this case).



```
Open  a.txt  Save
~/q2
1 Process id: 2
2 Name: kthreadd
3 Priority: 120
4 State: 1
5 Parent: 0
Ln 5, Col 10  INS
```

[The 5 fields specified earlier are also saved to the text file the user mentioned]

Error Handling:

Error	Where	Linked errno
If filename is not provided	Handled while taking input	61 (No data available)
If PID is negative	Handled while taking input	22 (Invalid Argument)
If PID doesn't exist	Handled if task_struct for PID points to null	22 (Invalid Argument)
If file cannot be created/opened/written to	Handled if appropriate permission not present	22 (Invalid Argument)
If string copying from user space to kernel space fails (unexpected page fault)	If unexpected page fault arises	14 (Bad Address)