Assignment 2. Part 2 WriteUp {Shashank Dargar: 2019107}

To test:
Download the C code and Makefile in the same directory and make sure the system is running on patched kernel. Use ./a.out <pid arg> to test syscall for given pid.

```
$ make
$./a.out <PID_argument>
<output>
```

Description of the Code:

To implement the System call in Linux kernel, in the source code two major patches were needed to be made:
The first one was in the syscall description table where a new entry was made, describing the syscall number, the name of syscall and the function which it will execute. These changes were made in the file **arch > x86 > entry > syscalls > syscall_64.tbl**. In my case, it was entry number 338 which was also reflected in the diff.txt file.
The second major edit in the source code was made as to the description (prototype and declaration) of the function in the file **kernel > sys.c**. The definition of the function is as follows:

```
SYSCALL_DEFINE1(sh_task_info, int, pid)
```

Here SYSCALL_DEFINE1 is a macro which is interpreted by the compiler as the name of system call is sh_task_info and it takes one argument, the argument being pid of int type. The function uses find_get_pid() to obtain pid struct and then the function pid_task() to get the task struct from the pid struct. The required fields are then printed to kernel logs using the function printk().The desired file is opened using the command filp_open() which takes the path and mode of the file as an argument (which is hard-coded). The values are then printed into the file using command kernel_write. The function is tested using a test.c code.

The Inputs User should Give:

The user should give the pid as an argument to the test.c code of the desired process of which the task_struct have to be accessed. The user should give only one input per execution.

Expected Outputs:

The following are the outputs:

<Fields Printed>
The system call returned with value 0.
>> This means successful returning of the code from the system call and the user is able to see desired fields both as output and also in kernel logs.

<Unable to open the file>
System call returned -2

>>This means that the test.txt file was not opened due to memory error. However output on kernel logs will also be discarded.

<Invalid Argument Entered. System call terminated.>

>> This implies that the argument entered is not an integer greater than zero hence can't be interpreted by the system call.

<The process with given {pid} does not exist. The system call returned with value -1. >

>> This implies that the system call was executed but was unable to find any process with given pid.

Errors values:

C function returning -1: Invalid Argument/ No argument passed. If the user tries to give a non-positive Integer as an argument so the C function will exit with value -1.

Syscall returning -1: It implies that the PID given by the user does not correspond to any of the running processes in the system.

Syscall returning -2: It implies that the system call was unable to open the desired file due to memory or kernel error.

Syscall returning 0: Successfully written to kernel logs.

C function returning 0: Successfully executed test.c function.