horizontal line

**Vrinda Singhal**

2018421

Linux System Call

**Assignment 1**

**Diff**

**Description of code and how I implemented the function:**

Implemented the system call sh\_task\_info which takes 2 arguments- unsigned int file descriptor an

Process PID.

SYSCALL\_DEFINE2(sh\_task\_info, pid\_t, PID, unsigned int, file\_descriptor)

First, it is checked if a file of given file descriptor exists by making a struct fd myfile and checking myfile.file

if not -1 errno is return

Then task\_list\_pointer to task\_struct is made which is used to loop over all the processes to match the pid,

task\_pid\_nr(task\_list\_pointer) == (int)(PID)

if no such process is found -2 errno is returned.

If the process is found, its details are printed with printk with changed priority of KERN\_DEBUG i.e. 7 so that it directly prints to the current terminal and not the kernel log file.

For writing to a file from the kernel space i made a function:

void Write\_to\_file(unsigned int fd, char\* buf, size\_t count)

Which works similar to the system call write.

Another helper function reverse and itoa is made to assist the population of the char buf[] the char array of content to be written in the file.

If the entire process of printing and writing occurs without an error 1 is returned else 0

And the system call breaks out of for\_each\_loop

**A sample C program to test out your implementation of the system call.**

**Sample input:**

**#define SYS\_sh\_task\_info 451**

**#include <stdio.h>**

**#include <linux/kernel.h>**

**#include <sys/syscall.h>**

**#include <errno.h>**

**#include <unistd.h>**

**#define SYS\_sh\_task\_info 451**

**void syscal(int pid, char\*\* file)**

**{ printf("-------Invoking Test For 'sh\_task\_info' System Call-------\n");**

**int fd=open(file);**

**int status = sh\_task\_info(pid, fd);**

**if(status == 1)**

**printf("System Call 'sh\_task\_info' executed correctly.\nReturn value 1\n");**

**else**

**{**

**printf("System call sh\_task\_info did not execute as expected\n");**

**perror("Error ");**

**printf("Error No.: %d\n", status);**

**}**

**}**

**int main()**

**{**

**syscal(); //pid and full path to the file**

**syscal();**

**syscal();**

**return 0;**

**}**

**Error no implication:**

-1 file not found

-2 pid not found

1- Normal execution