Operating System
Assignment 2
Simple System call

Kaustav Vats 2016048
Saksham Vohra 2016085

We defined our system call in the file sys.c, which is located in the /kernel/ folder present in the extracted folder of our kernel.
We defined the function in the following way : sys_definex(), where x is the number of parameters that are passed to the system call.
In our case, we defined it as sys_define2(sh_task_info,int,pid,char*,file_name), here x=2 since we have 2 parameters ie: pid and filename. The first parameter in the definition is compulsory and is the name of the system call implemented.

To retrieve the pid_struct of the task with the given pid, we used a function called find_get_pid, which returns struct pid, through which we can access the various data fields of a pid through the function pid_task which returns the task_struct which in turn contains all the information about the given process.

To write the data fields to the file provided by the user, we use the function sys_open ,sprintf and vfs_write, which are declared in the header files <fs/open.c> , <stdio.h> and <fs/read_write.c> respectively. sys_open is provided with the filename that the user has provided, and some flags depicting the conditions under which we need to write the file. We used the flags O_WRITE(open the file in write mode), O_CREAT(create the file if not already created), and O_TRUNC(if the file exists, then delete the existing contents of the file, else create it). We use vfs_write() to write a char* to the file and to write the non character items, we use sprintf() to write them to a character array provided to sprintf().

The function sys_open returns a file descriptor which depicts if there was an error while opening the file or not. If the file descriptor is 0, then the file was opened successfully. Else there is an error, and the file descriptor sets the value of errno defined in the '/usr/include/asm/errno.h' file to the error number describing the type of error, which we can later handle.

The user is expected to provide the syscall with a valid pid and a filename.

The output would be written in the file provided by the user, and the output would contain following details about task_struct:-

1) Name of the process
2) Pid number of the process
3) Name of the parent process
4) Pid number of parent process
5) State of the process
6) Priority of the process

Error values and how to interpret them.

Operating System
Assignment 2
Simple System call

Kaustav Vats 2016048
Saksham Vohra 2016085

Syscall returns -1 or 1 if any error. Else it returns 0 for success execution of syscall.

When a system request fails it returns a error code and these error codes are defined in errno.h file in this path '/usr/include/asm/errno.h'
ESRCH 3 /* No such process */
When we enter non existing pid.

ENOENT 2 /* No such file or directory */
When we enter wrong text file name or non compatible file name.

EINVAL 22 /* Invalid argument */
When we enter invalid arguments.