Hermann Krohn – 20057435
Elikem Hermon - 20075527

Lab 2 – Description and Pseudo-code of Program

Description:

The program outputs the number of processes and the number of threads. In addition, it outputs the PID, UID, and NICE for every running process. To accomplish this task, the program starts in the init_module function and creates an entry in the /proc filesystem using the create_proc_entry function. This function takes three parameters: the name of the file, the permissions, and the file's parent which is NULL in this case. If the create_proc_entry returns NULL, the program exits with an error. If there is no error, the program initializes the read_proc field with a function (my_read_proc) that is called every time the file is read.

The my_read_proc function checks if the offset (fpos) is zero which means that the program is at the top of the file. In this case, a pointer is initialized to obtain the value of the number of threads at a predetermined memory location (0xc038b3a8). Then, the headers are printed using sprintf. FirstTask is then updated to point to the location of init_task and lastTask points to firstTask. FirstTask is printed and lastTask is advanced to the next task. In the case that the offset (fpos) is not zero, the program checks if the firstTask is equal to the lastTask which signifies that all processes have been printed. If they are equal, a value of zero is returned to signify the end of the file. In addition, *eof is set to zero (end of request) and *start is set to page. If all processes have not been printed, the PID, UID, and NICE for the current task is printed and lastTask is advanced to the next task.

When the module is removed, the cleanup_module function is invoked which calls the remove_proc_entry function which removes the module.

Pseudo-code:

```
My_read_proc(buffer, start, fpos, blen, eof, data) {
        numChars = 0
        if fpos is zero {
                Initialize pointer to number of threads
                Write headers
                Update firstTask to address of initTask and lastTask to firstTask
                If firstTask processID not zero {
                        Print UID, PID, NICE for that process
                }
                Advance last task to the next task
        Else {
```

```
                If firstTask equals last task {
                        Set eof to 0
                        Set start to page
                        Return 0
                }
                If the PID of the lastTask is not 0 {
                        Print out its PID, UID, NICE
                }
                Advance lastTask to the next task
        }
        Set eof to 1
        Set start to page
        Return numChars
}

Init_module() {
        Create a proc_entry with the name lab2 and with 0444 for permissions
        If proc_entry is NULL {
                Remove the proc_entry
                Return -ENOMEM
        }
        Initialize proc_entry to use the my_read_proc function
        Return 0
}

Cleanup_module() {
        Remove the proc entry
}
```