Authored by: Adam Bayley and Samantha Hawco

ELEC 377 Lab 2 Description

In this laboratory, the kernel's internal workings were exposed. The kernel enforces security in user programs, protecting users from accessing restricted memory. The problem to be solved by this lab is the implementation of the *ps* command found in Linux. *ps* prints out the Process ID, virtual memory size, and resident set size. To accomplish this, functions were created and a kernel module was implemented through an output buffer called `numChars`. `the_read_function` which was used to coordinate the location of the current file, gathering the process information, and printing it. The module handled two cases, the start position being 0 and non-zero, using a circularly linked list. The process' PID, UID, VSZ, and RSS were sent to the buffer in `the_read_function` through the task structure. Some tasks utilized no virtual memory, resulting in mm values of NULL. The task structure iterated through each task, printing the process list while skipping values for which PID was 0. The program stopped looping when the process' `next_task` pointer pointed to the starting position pointer, `firstTask`. This was accomplished using a do-while loop.

For module creation and modification, a new proc entry was created using the `create_proc_entry` function. This was used in `init_module`, the function created to serve as the kernel's main. `init_module` was loaded using insmod and removed from memory after use by the `cleanup_module` function, which utilized rmmod. Specific difficulties faced in implementing the module and `the_read_function` can be found in the other documentation file, ELEC 377 Lab 2 Test.