# ELEC 377 Lab 1 Documentation

Gia Lee, Valerie So, William Robson

## Purpose:

This program is used to find and print out information about the current process. When running on the virtual machine, this program can give valuable information and insight about how the data the Kernel uses and how it handles a process.

The program runs the lab1_init method creates the lab1 process, (with check for failure), then it runs the main code in lab1_show to find and print out the desired values, then it uses lab1_exit to remove the process and finishes.

## Accessing Data Structures:

This code pulls values from two different structures (technically one, as one is within the other)

- "Task_Struct" Structure

Task_Struct is the data structure that holds all the process information for the current process for the kernel. It is accessible via the current-> pointer, which gives the information for the "current" process. This data is used in several ways for the OS to handle the process correctly.

The process name, PID, and State are found here.

- "Cred" Structure

The Cred (Credentials) structure is within the task_struct, and can be accessed via current->cred

It contains all the userID and groupID information used in the program.

## Problem Solution:

**Init and exit:**

```
77  static int __init lab1_init(void) {
78
79    if( proc_create("lab1", 0, NULL, &lab1_fops) == NULL){
80      return "ENOMEM";
81    }
82    //create lab1 file
83    printk(KERN_INFO "lab1mod in\n");
84    return 0;
85  }
86
87  static void __exit lab1_exit(void) {
88     remove_proc_entry("lab1", NULL);
89     //remove lab1 file
90    printk(KERN_INFO "lab1mod out\n");
91  }
```

-proc_create will return a null value if creation fails. If creation fails, program returns appropriate as error message "ENOMEM".

-remove_proc _entry removes the process that was created earlier on exit.

**proc_ops and file_operations**

-Used to set up the operating map depending on the current version.

**Lab1_Show**

-Uses seq_printf() to print values to console, values are found in task_struct using pointers.

-When using current->state, the state is stored as an integer, so there is an if/else structure to associate the integer with the proper state: running, waiting, or stopped.

-To get the PPID (the Process ID of the Parent) the command task_ppid_nr(current) is used to return the correct value, which is different from the other values which are just variables from the task_struct.