ELEC 377: Operating Systems

Lab 1: Process Information Kernel Module
Design Document
Bradley Stephen & Ben Wyand

1. Problem Description

The goal of this lab was to create a Linux kernel module that generates a file within the /proc directory, specifically named /proc/lab1. This file provides detailed information about the currently running process, such as its name, process ID (PID), parent process ID (PPID), current state, and the user/group IDs (UIDs/GIDs). The module interacts with the kernel's process control block (PCB) to retrieve and display this data, which helped us better understand how the Linux kernel handles processes under the hood.

2. Key Data Structures

- **struct task_struct**:
  - This is the core data structure used by the Linux kernel to represent processes. Each process running on the system is tied to an instance of this struct.
  - **Relevant fields**:
    - comm: Holds the name of the process.
    - pid: The process ID.
    - state: Indicates the current state of the process (like running or waiting).
    - cred: A pointer to struct cred that contains the process's user and group IDs.
    - parent: Points to the parent process, which we use to get the PPID.
- **struct cred**:
  - This structure stores the credentials of a process, particularly its user and group IDs.
  - **Relevant fields**:
    - uid: The real user ID.
    - euid: The effective user ID, important for access control.
    - suid: The saved user ID, useful in privilege management.
    - gid: The real group ID.
    - egid: The effective group ID.
    - sgid: The saved group ID.
- **struct proc_ops**:
  - This structure defines the set of operations we can perform on the /proc/lab1 file.

- **Relevant operations**:
    - proc_open: Triggered when the /proc/lab1 file is opened.
    - proc_read: Called when the file is read.
    - proc_lseek: Manages file seeking operations.
    - proc_release: Called when the file is closed.

3. Solution Approach

1. **Initialization (lab1_init)**:
    - When the module is loaded, it creates a new entry in the /proc directory called lab1. This is done through the proc_create() function, which registers the file and links it to the file operations described in the lab1_fops structure.
2. **File Operations**:
    - When the /proc/lab1 file is opened, the lab1_open function is invoked. This uses the single_open function to set up the file for reading and associates it with the lab1_show function, which formats the process information.
    - The lab1_show function pulls relevant data from task_struct (like the process name, PID, PPID, state, and UIDs/GIDs) and formats it using seq_printf so it can be displayed when the file is read.
3. **Cleanup (lab1_exit)**:
    - When the module is unloaded, the lab1_exit function is executed to clean up. It removes the /proc/lab1 file with remove_proc_entry(), ensuring that any resources allocated during the initialization phase are properly released.