ELEC 377: Operating Systems

Lab 1: Process Information Kernel Module

Testing Document

Bradley Stephen & Ben Wyand

1. Testing Environment

The tests were run in a controlled environment on a Linux virtual machine provided by the university's computer lab. The system had the required Linux kernel headers and tools installed, making it ready for compiling, loading, and testing kernel modules. All tests were carried out here to ensure consistency across the board.

2. Test Cases

Test Case 1: Compilation of the Kernel Module

- **Objective**: To confirm that the kernel module compiles without errors.
- **Procedure**:
    1. Navigate to the directory containing the source file (`lab1mod.c`) and the corresponding Makefile.
    2. Run the `make` command in the terminal.
- **Expected Outcome**: The module should compile without errors, generating a `lab1mod.ko` kernel object file.
- **Actual Outcome**: The module compiled successfully, and the `lab1mod.ko` file was created with no errors.

Test Case 2: Loading the Kernel Module

- **Objective**: To ensure that the module loads correctly and that the `/proc/lab1` file is created as expected.
- **Procedure**:
    1. Load the module by running `sudo insmod lab1mod.ko`.
    2. Verify that the `/proc/lab1` file exists by listing the contents of the `/proc` directory using `ls /proc/lab1`.
- **Expected Outcome**: The module should load without issues, and `/proc/lab1` should be present.
- **Actual Outcome**: The module loaded successfully, and the `/proc/lab1` file was correctly created.

Test Case 3: Verification of Process Information

- **Objective**: To check that the `/proc/lab1` file accurately displays process details, especially focusing on the UID and GID fields.
- **Procedure**:
    1. Open the `/proc/lab1` file with `cat /proc/lab1` and inspect the displayed process details, including process name, PID, PPID, state, and UIDs/GIDs.
    2. Cross-reference the UIDs and GIDs by checking the `/etc/passwd` file using `more /etc/passwd`. The `passwd` file stores user account information, including their UIDs and GIDs, which are essential for managing permissions.
    3. Repeat the test while logged in as the root user to compare the difference in UIDs and GIDs.
- **Expected Outcome**: The information in `/proc/lab1` should match the user's specific UIDs and GIDs as listed in `/etc/passwd`. When run as root, the UIDs and GIDs should display as 0.
- **Actual Outcome**:
    1. **For the regular user (netid: 19bbs2)**:

```
java
Copy code
Current Process PCB Information
Name = cat
PID = 15865
PPID = 15488
State = 0
Real UID = 1007
Effective UID = 1007
Saved UID = 1007
Real GID = 1000
Effective GID = 1000
Saved GID = 1000
```

The UIDs and GIDs matched the entries in `/etc/passwd`, confirming that the module correctly retrieves user-specific data.

    2. **For the root user**:

```
java
Copy code
Current Process PCB Information
Name = cat
PID = 16094
PPID = 15918
State = 0
Real UID = 0
Effective UID = 0
Saved UID = 0
Real GID = 0
Effective GID = 0
Saved GID = 0
```

As expected, the module returned 0 for both UIDs and GIDs when run as the root user, reflecting elevated privileges.

3. Test Summary

The testing process successfully verified the functionality of the kernel module developed for Lab 1. The module compiled, loaded, and displayed accurate process information within the `/proc/lab1` file. We used the `/etc/passwd` file as a reference for verifying UIDs and GIDs. This file is crucial because it stores essential information about user accounts on the system, including each user's unique User ID (UID) and Group ID (GID), which are critical for access control and permissions. By cross-referencing the values displayed by the module with those in `/etc/passwd`, we confirmed that the module correctly retrieves and reflects the assigned UIDs and GIDs for each user. Testing under both regular user and root conditions further confirmed the module's reliability in distinguishing user-specific information from root-level access, reinforcing its overall accuracy.