Lab Exercise 1: Kernel Compilation and Simple Modification

COP 4610 — Operating Systems Principles
Demo in Lab sessions: U01 - Tue Jan 31; U02: Fri Feb 3
Deliverables due on February 3, 2017

1. Description

The lab assignments for this course will involve modifications and additions to the *Linux* kernel. For most students, this is unfamiliar and intimidating territory. The goal of this project is to introduce you to the structure of the *Linux* operating system, the coding standards used in *Linux* kernel coding, and the development environment you will use for all of the lab assignments.

The first lab exercise is a warm-up exercise. You first need to create a new virtual machine (VM) using VirtualBox [2], which has a *Linux* OS (Ubuntu 10 [1]) installed. Then you will need to compile and install the kernel (version 2.6.36) in this *Linux* system. Next you will create/insert/remove a *Linux* kernel module. Finally, you will add a new system call in the *Linux* kernel. This lab exercise can be done on any workstation in ECS 141, ECS 241, or on your own laptop if you install VirtualBox on it.

For demo, the students should have their VMs ready before the lab session (3pm) on their scheduled lab session and the TA will grade their results during that session. Students using their own laptops should bring them to the TA for grading. Students who cannot attend that lab session should make an appointment with the TA for grading before 3pm on the demo day. Deliverables mentioned in Section 4 must be submitted in Moodle by 11:55pm on February 3rd. Late submissions will not be graded.

2. Requirements

- Virtual machine properly working
- The latest kernel successfully installed and running (2.6.36)
- The simple kernel module loaded and running
- Kernel 2.6.36 with the new system call working properly and passed the application test

3. Assignment

Note: The file references below are relative to the top of the Linux source tree (i.e., the linux-2.6.36-dev directory). The guides can be found in the “Documentation and useful links” area of Moodle.

3.1 Follow the *VirtualBox Guide* to create your virtual machine.

3.2 Follow the *Kernel Installation Guide* to configure, compile, and install (on your VM) the 2.6.36 kernel that you downloaded.

3.3 Modify, compile, install, and uninstall a Loadable Kernel Module (LKM).

- Start with the code and instructions in the *Kernel Module Guide*.
- Refer to this web page [3] if you have trouble with the sample Makefile.
- Add a header block to the “Hello World” sample code (e.g., test-toy.c) containing your name. E.g.,

```
/* John Smith */
/* COP 4610 Lab 1, Spring 2017 */
```

- Change the *printk* statements in hello_init so they print out your name. E.g., “Hello world, this is John Smith”. Make similar changes to hello_exit.
- Change the fields of the MODULE_AUTHOR and MODULE_DESCRIPTION macros appropriately.
- Compile your toy module into a loadable kernel module on your VM.
- Run ‘lsmod’. What do you see?
- Insert your module into the kernel by running ‘insmod test-toy.ko’.
- What message from your module do you see? Collect them. This will be part of your deliverables.
- Run ‘lsmod’ again. What do you see now?
- Unload your module.
- Run ‘lsmod’ again. What do you see now?

3.4 Create a new system call as described below. (Note: This involves modification of existing kernel code, so be careful and precise.)

- Follow the instructions in the *Implementing a System Call on Linux 2.6 for i386* Guide.

- Your new system call function takes one input parameter (an integer representing your panther ID). It should identify itself and then display the input parameter as well as the **system time** (in a human-readable format such as hh:mm:ss, month day, year). Name your function sys_xx_yy, replacing xx with your first name and yy with your last name, and provide a brief description of the function. Here's an example, showing how to do the first part of the requirement:

```
asmlinkage long sys_john_smith (int pantherid)
{
        printk ("sys_john_smith called from process %d with panther ID %d.\n",
                current->pid, pantherid);

        /* Put your code to display the current time here*/

        return 0;
}
```

  Note: you need to figure out how to get the system time in the kernel. **Hint**: refer to Chapter 7 of Linux Device Drivers [5], and this manual page [6].

- Recompile the kernel. Verify that your system call is in System.map (use *grep*).
- Install the modified kernel. Cross your fingers, and reboot. If the system reboots properly, your new system call will be available.
- Test your system call with a short program. You can adapt the example code in Step 15 of reference [4].
- Expected output: at the user level, you should see ‘0’; at the kernel level, you should see “sys_john_smith called from process …”
- If you don't see the expected output on the screen, look at the tail of *dmesg*.

4. Deliverables

- Source code for your loadable module in Section 3.3, as well as the output information that you collected.
- A patch file that represents the changes between the vanilla 2.6.36 kernel and the 2.6.36-dev kernel

with your modification in Section 3.4. Follow the *Kernel Patch Guide* in Moodle to obtain this patch file and name it **my_syscall-xx-yy.patch**, replacing xx with your first name and yy with your last name.

- Source code of the test program for your new system call
- A report about what you learned from this lab assignment.
- Compress all the above into a single file and upload it into Moodle.

5. Grading criteria

- Ubuntu running and Vanilla 2.6.36 kernel compiled, installed, and can be rebooted into **(30%)**
- LKM can be installed and removed, and it displays the expected messages in the kernel log (dmesg output) **(+20%)**
- The new system call added and can be invoked **(+20%)**
- The new system call can show the current time **(+10%)**
- The new system call can show the current time in a human-readable format **(+10%)**
- Report **(+10%)**

6. References

[1] https://help.ubuntu.com
[2] http://www.virtualbox.org
[3] http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/
[4] Implementing a System Call on Linux 2.6 for i386: http://tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/
[5] http://lwn.net/Kernel/LDD3/
[6] http://linux.die.net/man/2/gettimeofday