CSCE 3402 Spring 2021
Exercise 8 – The Proc Filesystem
Kernel Module Integration with Proc System Call

Assigned: Tuesday, March 30th in Lab

Due: Tuesday, April 6th at Lab time
Delayed submission with penalty until Thursday, April 8th at 11:55pm.

Goals
The goal of this lab exercise is to extend your kernel module you developed in the previous exercises and integrate it with the **procfs**. The **procfs** is one of the convenient means for communicating with the kernel land from the user mode without having to use a special API.

Details
This lab exercise is an individual exercise that you need to carry out on your own. After you have managed to hook the fork system call and keep a count of the number of forks starting from the point when your kernel module is loaded into the kernel, it is now time to make this accumulated counter available on demand to the user mode.

The **procfs** is an in-memory virtual filesystem that acts as a system dump. Essentially, entries inside the **procfs** are event-driven returning important information that is usually stored in internal kernel data structures. The **procfs** extends the **VFS** mechanisms but on file-basis rather than a whole filesystem. Essentially, a **procfs** file entry is assigned a set of routines for different operations that can be performed on it; open, read, write, etc.

Your mission is to amend your kernel module to create a **procfs** entry upon loading named **"/proc/fork_count"** to be responsible for a very simple task which is to return the value of the fork counter maintained by you kernel module. You are required to create, and setup all needed data structures to create a **procfs** file entry and essentially provide an open and read routines at the minimum and plug them into the needed **procfs** data structure before registering it to the **procfs**. It is very important to mention that some operations might need to be used from the procfs default library without modifications; e.g. **seq_lseek**. Moreover, you need to remove the added profs entry upon your kernel module exit, and to do all house keeping needed to restore the system to its original state before inserting your kernel module.

Finally, after inserting your kernel module you should get the fork_count value upon executing the shell command "**cat /proc/fork_count**".

IMPORTANT Note: You need to make your kernel module work with the KASLR enabled.

What to submit

1. All the C code you wrote for the version kernel module.
2. You Makefile.
3. A small readme file explaining how to use your make files to compile the programs.

How to submit:

Compress all your work: source code, report, readme file, and any extra information into a zip archive. You should name your archive in the specific format <Student_ID>_<Name>_Lab8.zip. Finally, upload your code to blackboard.

Grade

This Lab exercise is worth 10 % of the overall course grade. The exercise will be graded on a 100% grade scale, and then will be scaled down to the 10% its worth. The grading of the assignment will be broken down as follows:

1. 10 % for just submitting a meaningful assignment before or on the due date. This 10% does not account for the correctness of your assignment but submitting an empty assignment without code will definitely results in loosing this 10% and consequently the whole grade of this assignment.
2. 80 % for the correctness and the quality of the submitted code and make files.
3. 10 % readme file.

Delays

You have up to 2 working days of delay, after which the assignment will not be accepted and your grade in that case will be ZERO. For every day (of the 2 allowed days), a penalty of 10% will be deducted from the grade. And of course you will lose the 10% mentioned in point 1 above under the “Grade” section.