CSCE 3402 Spring 2021
Exercise 5 – System Call Table Kernel Module
Kernel Module to Retrieve System Call Table Address and the Fork System Call Address

Assigned: Tuesday, March 9th in Lab

Due: Tuesday, March 16th at Lab time

Delayed submission with penalty until Thursday, March 18th at 11:55pm.

Goals

The goal of this lab exercise is to extend your kernel module you developed in the previous exercises and retrieve the System Call Table address. Further you should retrieve the **fork** system call address and print both on the screen. It is very essential to do this step to be able to do the next lab exercise.

Details

This lab exercise is an individual exercise that you need to carryout on your own. After you have managed to get the running kernel version, you should use that to access the correct System Map file from under the "/boot" directory. The file is named "System.map" followed by the version number you were able to retrieve in the previous lab exercise. You can list the directory "/boot" to learn more about the different kernel file names. You need to do that as you can have more than one kernel installed on your hardware and hence more than System.map file with a different version suffix, and you need to open the corresponding Systems Map file to your current running kernel.

So, what is System.map? :) It is a file that is generated by the gcc linker tiools in the kernel linking stage that has all kernel symbols and their expected corresponding addresses that they should be loaded at in the memory during runtime. You need to work your muscles a bit to use what you have so far to open the file and traverse it searching for the system call table address; the system call table is in the form of a variable instantiated from sum type of struct, which means that it is name is a symbol :) You need to figure out the name of the pointer variable representing the system call table.

Once you have that, you need to work your muscles even a bit more to locate the entry in the system call table corresponding to the **fork** system call and retrieve its content which stores the address of the sys_fork function :)

If you can do all the above, then you are ready for the next lab exercise :) and if you are struggling with it :(keep on trying hard until you get it done.

IMPORTANT Note: From now on please make sure that you boot up with the kernel grub entry you created in the lab exercise #3 that has the KASLR disabled.

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>_Lab5.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.