Brandon Patton

CS 492 Operating Systems

Section A

Github repo name: lab-3-linux-kernel-modules-bpattonstevens

Lab 3: Kernel Modules and Device Drivers

I. Purpose

The purpose of this assignment was to design and make use of kernel modules that write data to the kernel log when installed to the kernel and used. The two files we were asked to develop were hello.c and qmod.c, both of which were installed to the kernel using the command insmod and remove from the kernel using rmmod. We were also asked to take a video of each module being used and the output generated from using them. For the output, I went to the directory /var/log and put in the command tail -f kern.log to get the most recent end of the kernel log to find the hello.c kernel module output. Also, for each module, we were asked to use the command modinfo to show our names attached to each kernel module.

II. hello.c

This kernel module displayed greetings to the inputted name in the code. In this case, this program greeted “bpatton” on being loaded and said goodbye to “bpatton” when unloaded from the kernel. I made use of the pr\_info print function to print to the kernel log instead of the normal printk which requires the use of a flag to indicate where to print to (we have used KERN\_INFO in the past). As mentioned above, the message was able to be seen in the kernel log by navigating to the /var/log directory and inputting the command tail -f kern.log.

III. qmod.c

This kernel module used a shared queue data structure to store and share messages via a character device. Multiple messages are able to be added and removed by use of the commands: ./qadd <message> and ./qremove. By making the queue instance of the shared queue data structure a pointer and as a linked list, the queue instance was able to be shared throughout the program and added to and removed by dev\_write and dev\_read respectively. Each of the latter functions made use of the various data structures provided at the beginning of the file and made use of mutexes to ensure that race conditions did not occur. The programs themselves followed the necessary queue iterations and processes needed to remove and add messages from the queue by use of the attributes last, front, next. I also made use of a temporary variable called tmp and a current variable called curr to keep track of the current message being handled and to push along the queue.

IV. Difficulties

Throughout this lab, I encountered some difficulties, some of which were and still are mysterious in nature. For example, for qmod.c I was consistently receiving the error “Invalid Storage Class for <function name” for all the functions provided in the spec. Although this error was mentioned on the Piazza forum by another student, the solution provided in the comments did not apply to my document as I checked many times for declarations after execution of code. Strangely enough, as soon as I recopied all of my code to a fresh spec, this error did not occur despite being exactly the same structure as the initial document. Also, I encountered several null pointer dereference errors which had to do with how I was structuring the enqueuing an dequeuing process. I did not encounter any errors with hello.c.