CS409

Learning Experience

002269904

Liam Buck-T.

Task 1

For task 1, I learned to implement the proc file system, specifically how to read from a proc file, by making kernel modules that would create proc files for evaluating the length of time since a kernel module was loaded as well as the current value of jiffies. One issue that was encountered was the command “cat /proc/jiffies” that would simply display an error message “bad address”. This was due to trying to access a value within the proc_read() function without transferring it to user space. The solution to resolving the error was to use the function copy_to_user() along with a buffer that held the string value of the current jiffies. After doing so, the value was properly read within the proc file system and displayed in the terminal. To determine how much time had elapsed since a kernel module was loaded, a starting value of jiffies was obtained when the module was first loaded and then the current value of jiffies is then obtained when the command “cat /proc/seconds” is entered into the terminal where the starting value is subtracted from the current value and subsequently divided by the HZ rate to obtain the elapsed time in seconds.

Task 2

For task 2, I learned to write to the proc file system by taking advantage of the copy_to_user() function again, however this time it was used to take what was written to proc/pid to copy the written content into kernel memory. The function that allowed us to assign kernel memory was the kmalloc() function which enabled the use of the copy_to_user() function in copying the written pid. An experience with efficient memory allocation practices was introduced through the use of the kfree() function to release the allocated kernel memory. An issue arose when trying to access the state of the current pid where we could not find a way to properly display the associated state of the current pid. To get around this issue, I looked at the header file for sched.h and got the correct field name for the state under struct_task. After doing so, the proper state of the process was able to be read through the proc_read function after confirming the correct field name was “__state”.