**NAME OF PROJECT:**

================

Kernel Module and Processes

**MEMBERS:**

========

Troy Joslyn

Sherman Kettner

**STATEMENT:**

==========

We have neither given nor received unauthorized assistance on this work.

**VIRTUAL MACHINE (VM) INFORMATION:**

=================================

NAME OF VIRTUAL MACHINE (VM): Group 6 VM

USERNAME: root

PASSWORD: CS\_4500\_001

**Describe where the files can be found.**

VM Location: CS 4500 (Operating Systems) > skettner > Group 6 VM

**Describe each file and the purpose it serves.**

Print\_self.c: This file is used to print the child and parent processes of the kernel from the start of it to init.

Print\_other.c: This file is used to print the child and parent processes of the kernel based on an inputted pid that relates to the child and parent processes.

**Provide any special instructions to access or run your program.**

All the files are located in the CS-4500 folder. So, from the home directory (in the VM), cd CS-4500, then cd Project2, then cd into each part that needs to be looked at. There should be 4 parts (Part1, Part2, Part3, Part4) in that Project 2 folder.

**ROOT PASSWORD:**

==============

CS\_4500\_001

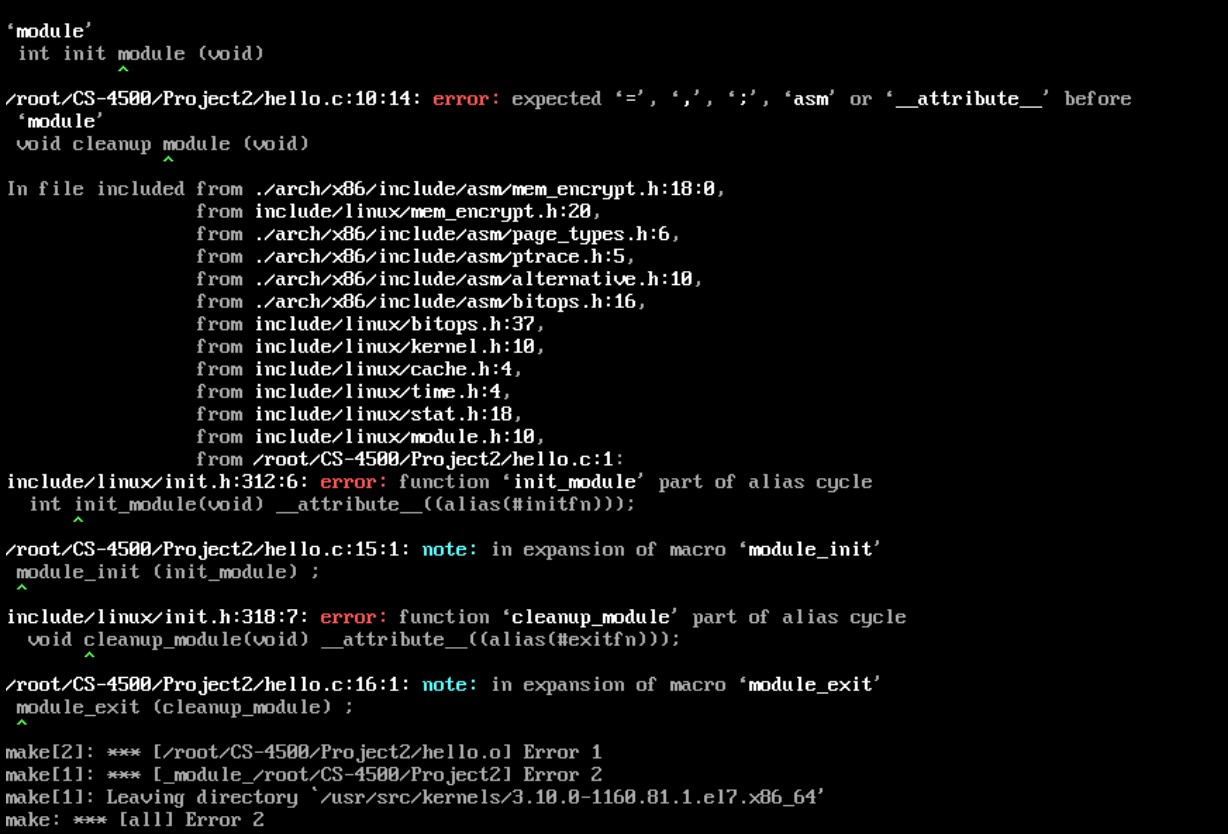
**DESCRIPTION AND REMEDIATION:**

============================

**Part 1:**

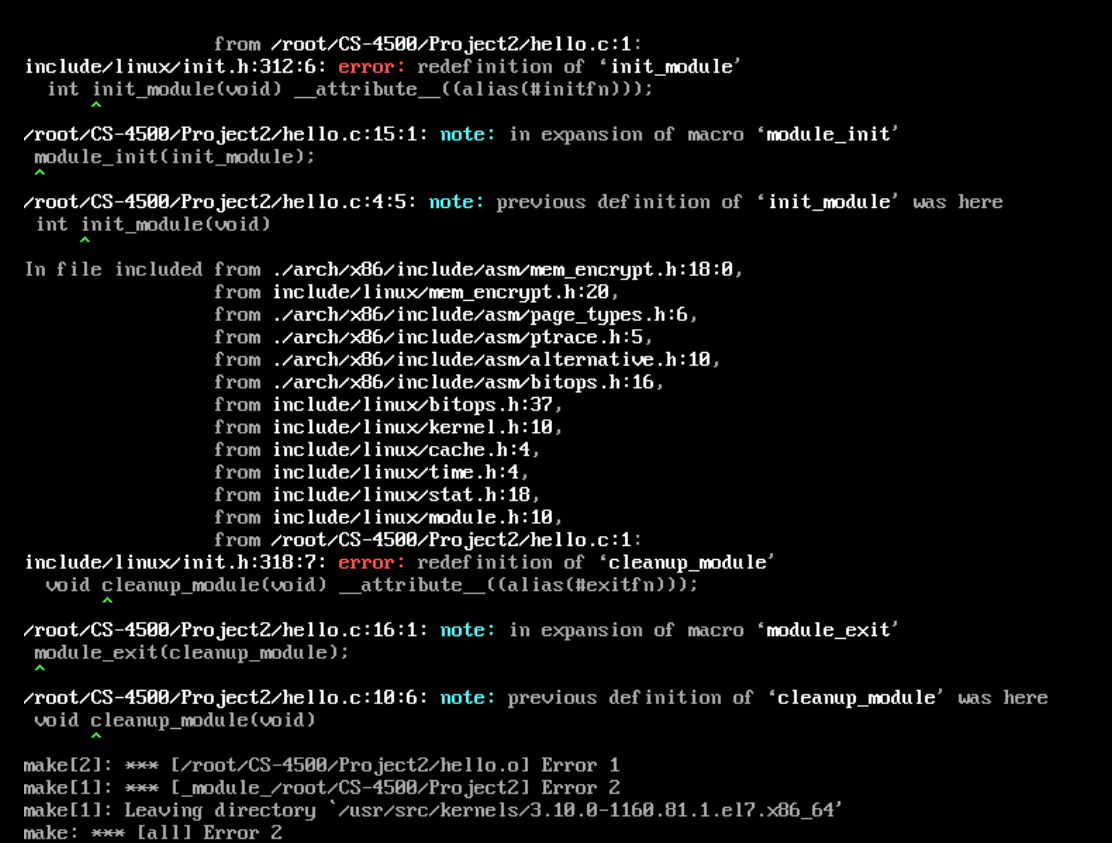
Answers to questions:

Yes, we see multiple errors when we make. Those errors are displayed below:



**To fix these errors we**

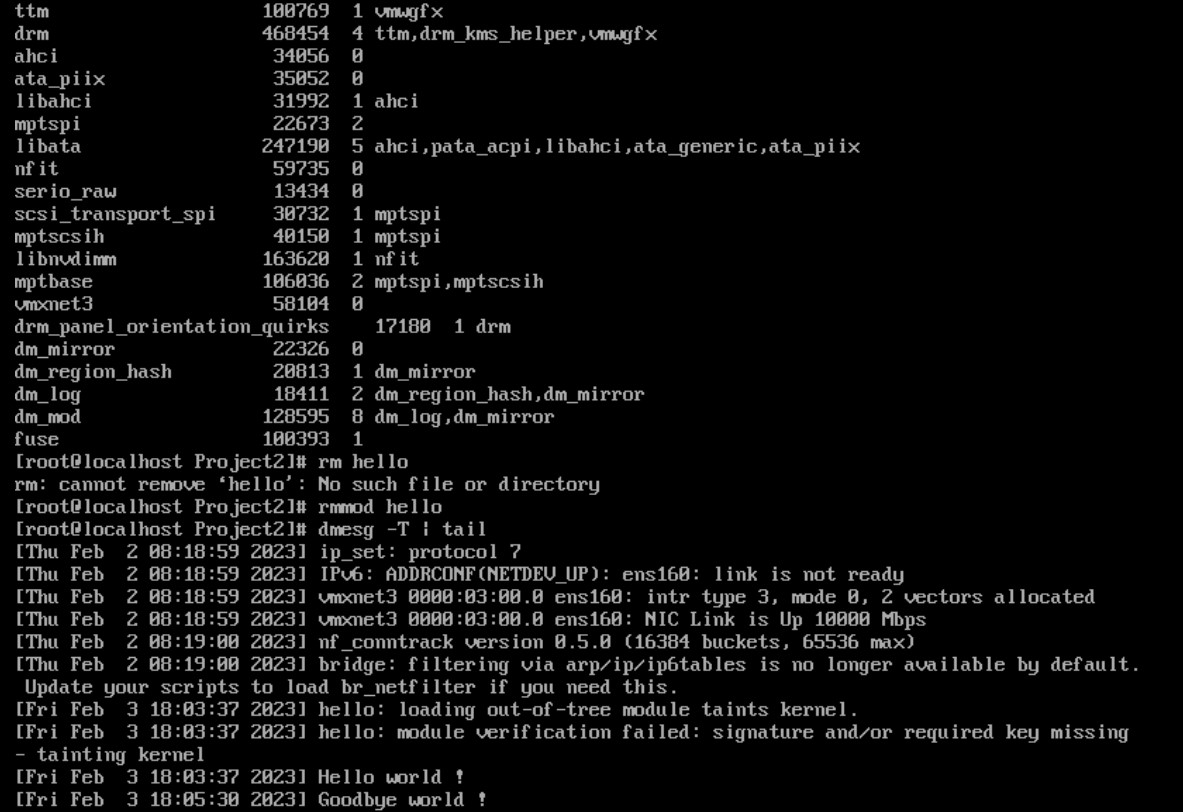
The first error that was fixed was the expected \_ before module because we missed the \_, once that was fixed the error were as followed.



These errors appeared to me to be a conflict of names so I simply changed the names, which solved the errors

**Notice the different timestamps when Hello world! and Goodbye world! are printed.**

**Timestamps Question:** Then, when we got it to work, we got Hello World and Goodbye World to print. The timestamps were almost 2 minutes apart from when they printed Hello World and Goodbye World. The example is below:



**If you want to print messages from init\_module and cleanup\_module to output you will change**

**Standard Output Question:** You can use pr\_info to also print data to the console (kernel) in the modules. However, if you are strictly talking about C, one of the only ways to print to the console would be printf. Depending on the application of the printing you can use for kernel printk, pr\_info or use printf if the data is in C.

**Part 2:**

*Steps to load and remove the module: insmod print\_self.ko and to remove the module do: rmmod print\_self. Then you can use dmesg -T to get the details the file is putting out.*

Answers to questions:

1. When the module is loaded the child process is the currently running process. - **insmod**
2. As discussed in our lecture, in old kernels the mother of all processes is called init. In newer kernels, what is it called and what do you see from your module’s output? - **systemd**
3. When printing state in your code, please map the numeric state to its string state, e.g., print TASK RUNNING if state is 0. From your module’s output, which state(s) are observed? - **TASK\_RUNNING, TASK\_INTERRUPTIBLE, and TASK\_STOPPED**

**Part 3:**

*Steps to load and remove the module: insmod print\_other.ko and to remove the module do: rmmod print\_other. Then you can use dmesg -T to get the details the file is putting out.*

**Part 4:**

Answers to questions:

1. The difference between a kernel module and system call is that a system call is a way to get into the kernel implemented with software generally using interrupts. It is basically a call to the kernel from the user to access a function or other item. The kernel module is the part of the OS which deals with process management, i/o, memory, etc. When a system call occurs the kernel executes the operation for the user and is the main part of the system in the OS.
2. The code does not work. This could be multiple issues mostly because it ended in a previous kernel version (support for the sys\_call). The first issue could be because we don’t have one of the .h files in the kernel. So, that could be an issue where they removed or changed functionality for the kernel throughout the years. This may possibly be a good thing because they would be updating the kernel to make it more secure and less vulnerable to outside entities trying to get in. If this file didn’t need to be included or had issues, it’s best to leave it out and find a work around to make the kernel more secure.

In theory, this program is supposed to interrupt something from the kernel so that it doesn't happen. For example, if you had a file you could interrupt the opening of the file if it is running in the kernel. In our case, it is trying to interrupt by making a fake exit call instead of the actual exit call.