Laboratory Exercise #5

Introduction to Kernel Modules on Zynq Linux System

Name: Daniel Horan
UIN: 527005307

ECEN 449-504

I. Introduction

In this lab, we aim to build and load a simple 'Hello World!' kernel module on the CentOS 7 workstations, targeted for the ZYBO Z7-10 board. Following this, an advanced kernel module will be created to not only display messages in the kernel's message buffer but also moderate kernel access to the multiplication peripheral from Lab 3. This exercise advances our understanding of kernel module development and its role in managing hardware-software interactions on FPGA platforms. Through this lab, we will further elucidate the interaction between kernel module programming and hardware peripheral integration in microprocessor systems.

II. Procedure

The exercise commences with the creation of a *'Hello World!'* kernel module on CentOS 7 workstations. This module is then loaded into the Linux kernel on the ZYBO Z7-10 board to establish a foundational understanding of kernel module interactions with the Linux kernel.

Post the initial module creation, attention shifts to developing an advanced kernel module. This module is engineered to print messages to the kernel's message buffer while moderating kernel access to the multiplication peripheral established in Lab 3.

Key steps in this procedure include booting Linux on the ZYBO Z7-10 board, and mounting the SD card for read and write access using the 'PICOCOM' on the CentOS machine for command execution. After these preliminary steps, the *'petalinux-create'* and *'petalinux-build'* commands are utilized for module creation and building, respectively. Upon successful compilation, the *'hello.ko'* file is generated and transferred to the SD card.

The SD card is re-inserted into the ZYBO Z7-10 board, and the *'insmod /mnt/hello.ko'* command is executed to load the *'hello'* module into the Linux kernel. The output of the *'printk'* statement, activated when the kernel module is loaded, is viewed by executing the *'dmesg | tail'* command in the terminal window.

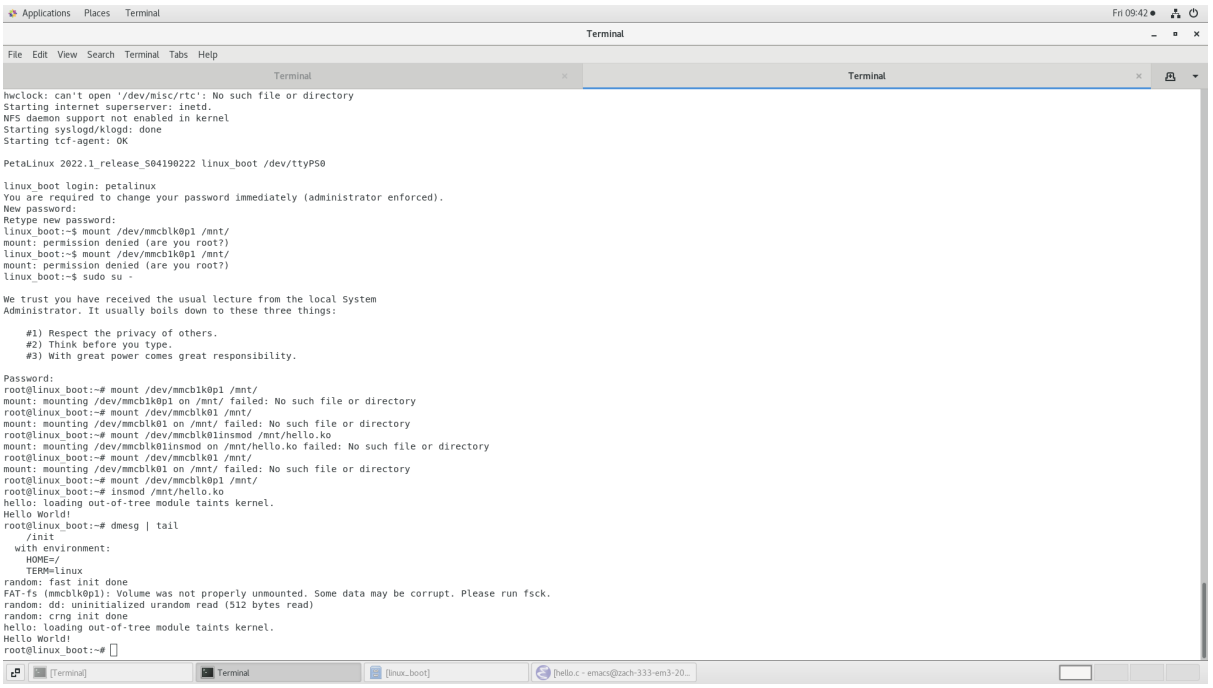
A parallel procedure is followed for the 'multiply' kernel module, focusing on interaction with the multiplication peripheral and logging the results to the kernel message buffer. Necessary code adjustments are made to the *'multiply.c'* file, followed by building the *'multiply'* module, transferring the *'multiply.ko'* module onto the SD card, re-inserting the SD card into the ZYBO Z7-10 board, and executing the *'insmod'* command to load the *'multiply.ko'* kernel module into the Linux kernel on the ZYBO Z7-10 board

III. Results

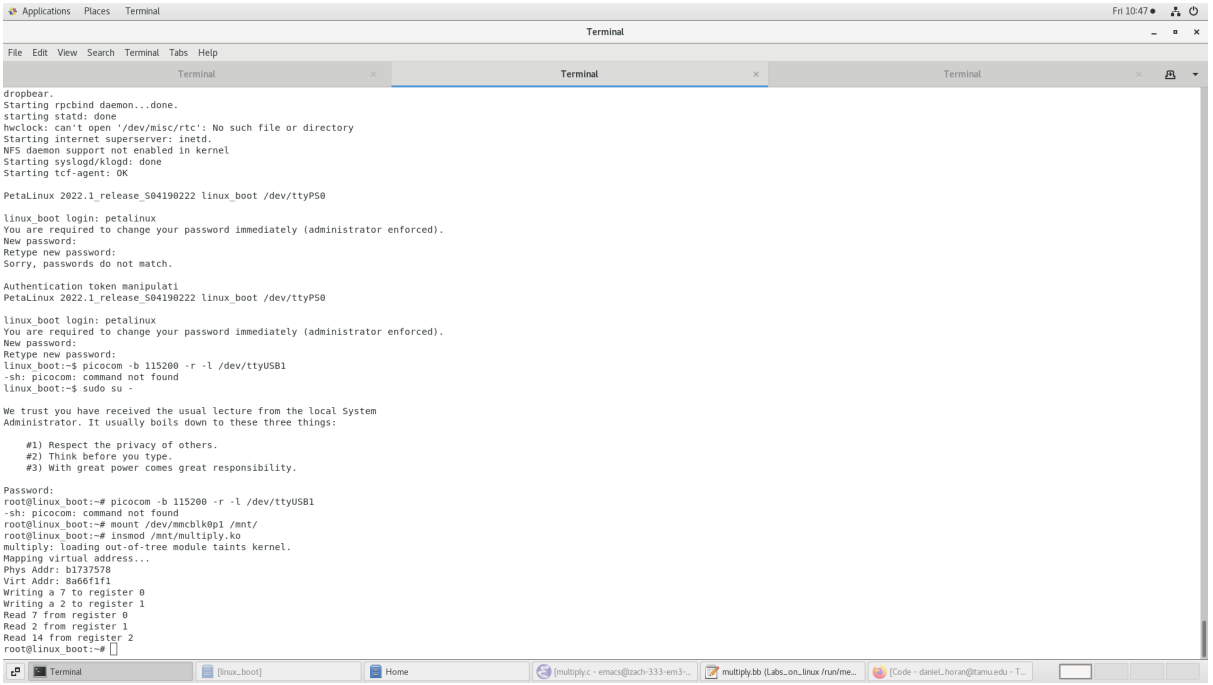
The primary aim of this lab was to create and load kernel modules on the CentOS 7 workstations, and then onto the Linux kernel on the ZYBO Z7-10 board. Initial efforts focused on a simple *'Hello World!'* module, which was executed seamlessly. This module served as a stepping stone to understanding kernel module interactions within the Linux kernel environment.

Transitioning to a more complex module, minor difficulties arose, particularly in selecting the right directories and adjusting the *'multiply.c'* file. Despite these initial hurdles, with meticulous attention to detail, the module aimed at moderating kernel access to the multiplication peripheral was successfully created and loaded.

Upon completion, a demonstration was carried out for the TA, showcasing the successful interaction between the kernel modules and the multiplication peripheral. Both the 'hello' and 'multiply' kernel modules operated as intended, validating the objectives of the lab. The minor issues encountered provided valuable insights into the importance of precise directory management and file modification in kernel module programming.



Screenshot 1: *'Hello World!'*



Screenshot 2: *'multiply.c'*

IV. Conclusion

This lab involved developing and loading kernel modules on CentOS 7 workstations for the ZYBO Z7-10 board, progressing from a basic 'Hello World!' module to a more advanced module interfacing with a multiplication peripheral. This transition highlighted the nuances of kernel module programming and its critical role in managing hardware-software interaction on FPGA platforms. The initial module laid the groundwork for understanding kernel module interactions within the Linux kernel environment. Moving to the advanced module introduced minor challenges in directory selection and 'multiply.c' file modification, underlining the importance of precise directory management and file modification. The successful demonstration to the TA affirmed the achievement of the lab objectives, showcasing the functioning 'hello' and 'multiply' kernel modules with the multiplication peripheral. This exercise not only met the lab's goals but also enhanced the comprehension of kernel module programming and its influence on hardware peripheral integration, fortifying the practical application of theoretical concepts in microprocessor system design.

V. Questions

a) If the ZYBO Z7-10 board is reset before step 2.f, in step 2.g, you'd need to re-boot Linux on the board and re-mount the SD card using the following commands in the PICOCOM on the CentOS machine:

$ sudo su - (enter password as ‘root’)
$ mount /dev/mmcblk0p1 /mnt/

Once the environment is restored and the SD card is mounted, you can proceed with creating the 'multiply' module as originally outlined in step 2.g.

b) The mount point is */media/daniel_horan*

c) If the name of hello.c is changed, update the Makefile to reflect the new file name for correct compilation.

If the kernel directory from lab 4 is specified in the Makefile instead of lab 5, it could lead to compilation errors, incorrect build behavior, or module malfunction due to possible differences in kernel configurations between the labs.