ECEN 449 Lab 5: Introduction to Kernel Modules on Zynq Linux System

02/27/2020

Jason Gilman

**Introduction:**

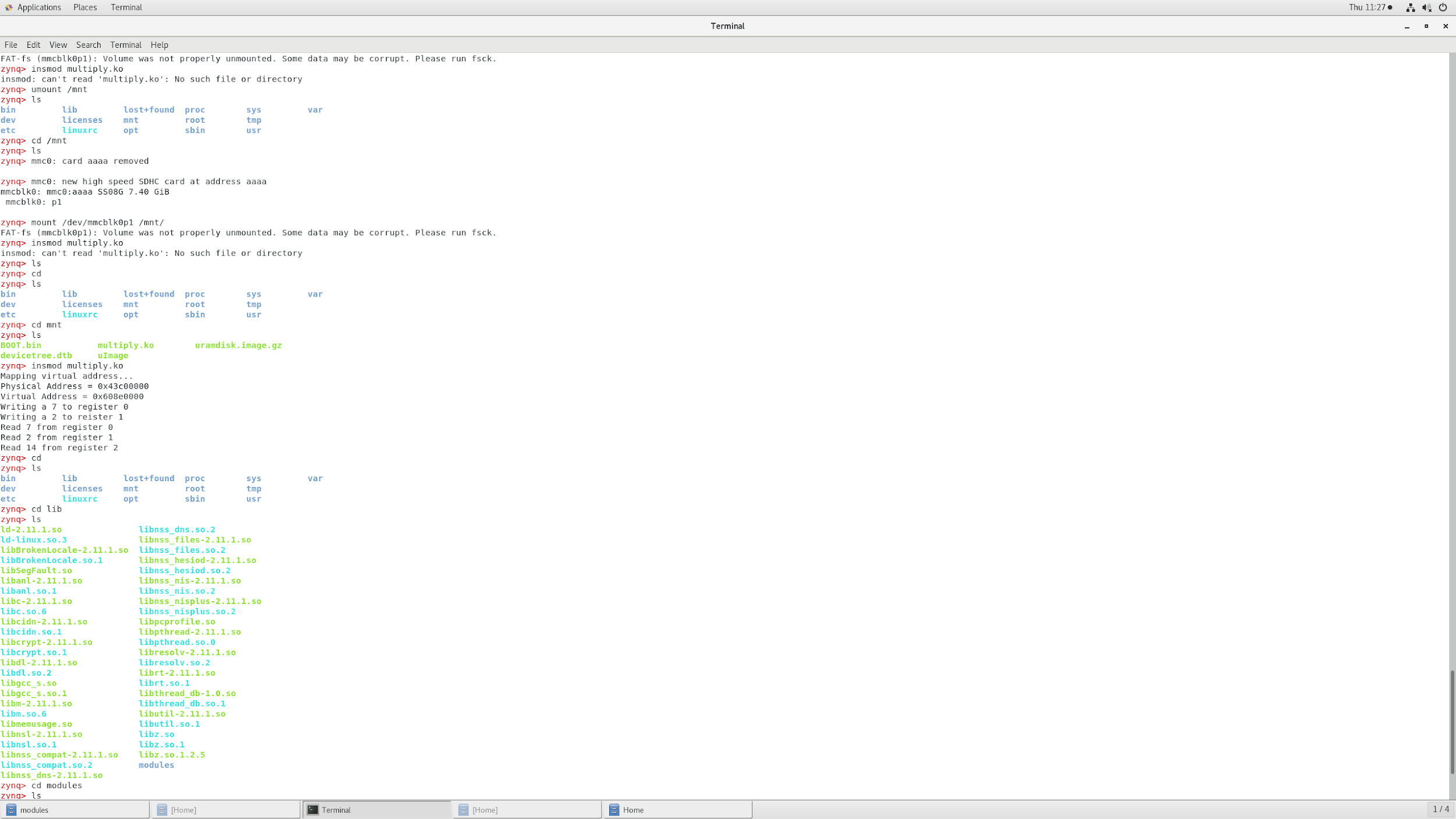
In this lab, we were tasked with creating a kernel module that will run on the linux operating system. The operating system will run on the Zybo-10 FPGA system. The output of the kernel module can be observed in the linux terminal using the picocom command.

**Procedure:**

* To start the process of lab 5, the linux operating system must be run on the Zybo-10. This involves loading the required files onto the sd-card, and then booting the system off of the kernel that is contained on the sd-card.
* The linux operating system can be interacted with by using the picocom command in the terminal.
* After exploring the linux operating system by looking through the file system, the next thing to do was to create a kernel module. This needs to be done on the cent-OS system, so the sd-card must be dismounted from the Zybo-10 system by using the “umount /mnt” command.
* On the cent-OS system, a module was developed by using c-code. The “hello.c” code contained an initialize function that writes “Hello world!” to the kernel output. There was also an exit function that outputs “Goodbye world!” to the kernel.
* The “hello.c” code was then compiled using a Makefile, and specific parameters to create a kernel module that could then be loaded into the linux operating system.
* Once again, the sd-card was prepared with the relevant files, and the drive was mounted on the linux OS. In order to run the kernel module that was just created, the command “insmod <module>” must be run within the picocom terminal.
* This would output the messages described above.
* Next, a new kernel module must be made. In this particular module, two numbers will be written to two separate registers using “iowrite32( num, addr)”. Then, the two registers will be read, and the product of the two will be stored in a third register. All three registers will be read, and outputted to the kernel output.
* The module is then loaded onto the sd-card, and then run on the linux os.
* The output, like the first module, is then outputed into the picocom terminal.

**Result/Output:**

The following was the output of the multiply kernel module that was run on the linux-os by using the picocom command. You can see the address in physical and virtual space, as well as the the integers that are contained in the first three registers. Register 0 and register 1 were multiplied, and the product is stored in register 2.



**Conclusion:**

By developing c code and compiling it using specific parameters relevant to the linux installation, we were able to create a working multiplying kernel module that was able to be run on the Zybo-10 by loading the module into the operating system. The output was observed in the picocom terminal.

**Post Lab Questions:**

1. Prior to step 2, we created a directory, test, which would disappear if the zybo was reset. Whenever the sd card was inserted back into the zybo-10, the drive would then need to be mounted again.
2. The mount point on the Cent-OS machine is under /media/jasongilman/<sd card device>
3. If the name of the c code file was changed, say <name>.c, you would have to find every instance of hello.o and change it to <name>.o. As far as our makefile goes, this should only be at the heading of the makefile, where “obj-m += ------” is contained. If the lab4 directory contained different linux setup files than contained in lab5, the module would not run correctly, although in my case, I used the lab4 linux directory without issue.

**Appendix:**

Multiply.c

