**Lab 7: Built-in Modules**

Alana Ordonez

UIN: 428008825

ECEN 449 506

December 2, 2022

**I. Introduction**

This lab familiarizes us with built-in kernel drivers to load during our boot-up of our Linux kernel. We use the multiplier driver from lab 6 and examine other built in drivers, learning how to enable and disable drivers/services, which leans the kernel image size and eliminates unnecessary functionality. We also had experience in creating a Kconfig file, and we honed our skills on writing our own makefiles.

**II. Procedure**

The first part of the lab was getting accustomed to the menu configuration, or menuconfig, of the Linux kernel. This allows us to view which characteristics and support capabilities are enabled for our current kernel, which we have built in previous labs. We then created a makefile for the multiplier driver that allows us to compile the source code executable for the driver. A Kconfig file was written for the multiplier driver as well, where we named the driver for the menu configuration, and specified which architecture we are using (ARM), along with a help directive for the entry. Then, we added the configuration line for the driver in the Linux source code’s makefile so that it configures it, too. Within this Linux source code folder, we also edited the Kconfig file to source the driver to make it accessible for the kernel. Once this was done, we selected the multiplier driver to be compiled as built-in and compiled it to generate a uImage. With this uImage, the Linux kernel was booted once again on our ZYBO Z7-10 board, where we can use the multiplier driver to execute our previously created module tester, “devtest.” Afterwards, we disabled the network device support, multimedia support, and soundcard support from the device drivers of the kernel’s menu configuration.

**III. Results**

After booting the Linux kernel, the terminal showed the printk() messages that were written for the multiply peripheral in our previous labs, showing that it was properly instantiated. The initial uImage size was 3.29 MiB, as shown in Figure 3. Figure 5 shows the uImage size after the three drivers were disabled, which was 2.96 MiB, showing that reducing the number of drivers/modules reduces the uImage size.

**IV. Conclusion**

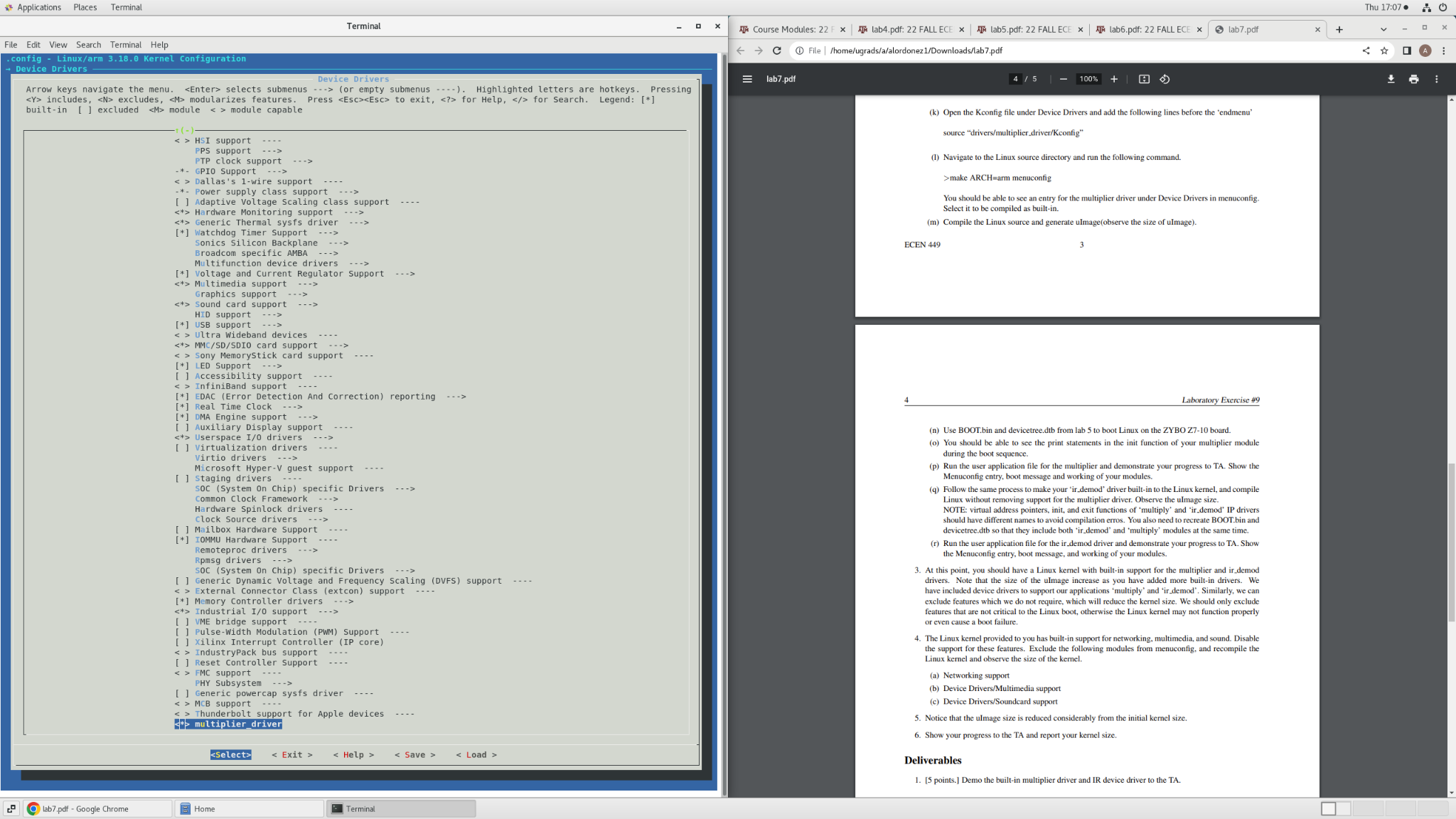
As the last lab, it was a great experience using peripherals and methods that have built off of each other over the course of the semester. The lab allowed us to get acclimated to configuring our Linux kernel to have only what we require, adding and removing drivers, and understanding how built-in modules and loadable modules can have varying time and size.

**V. Questions**

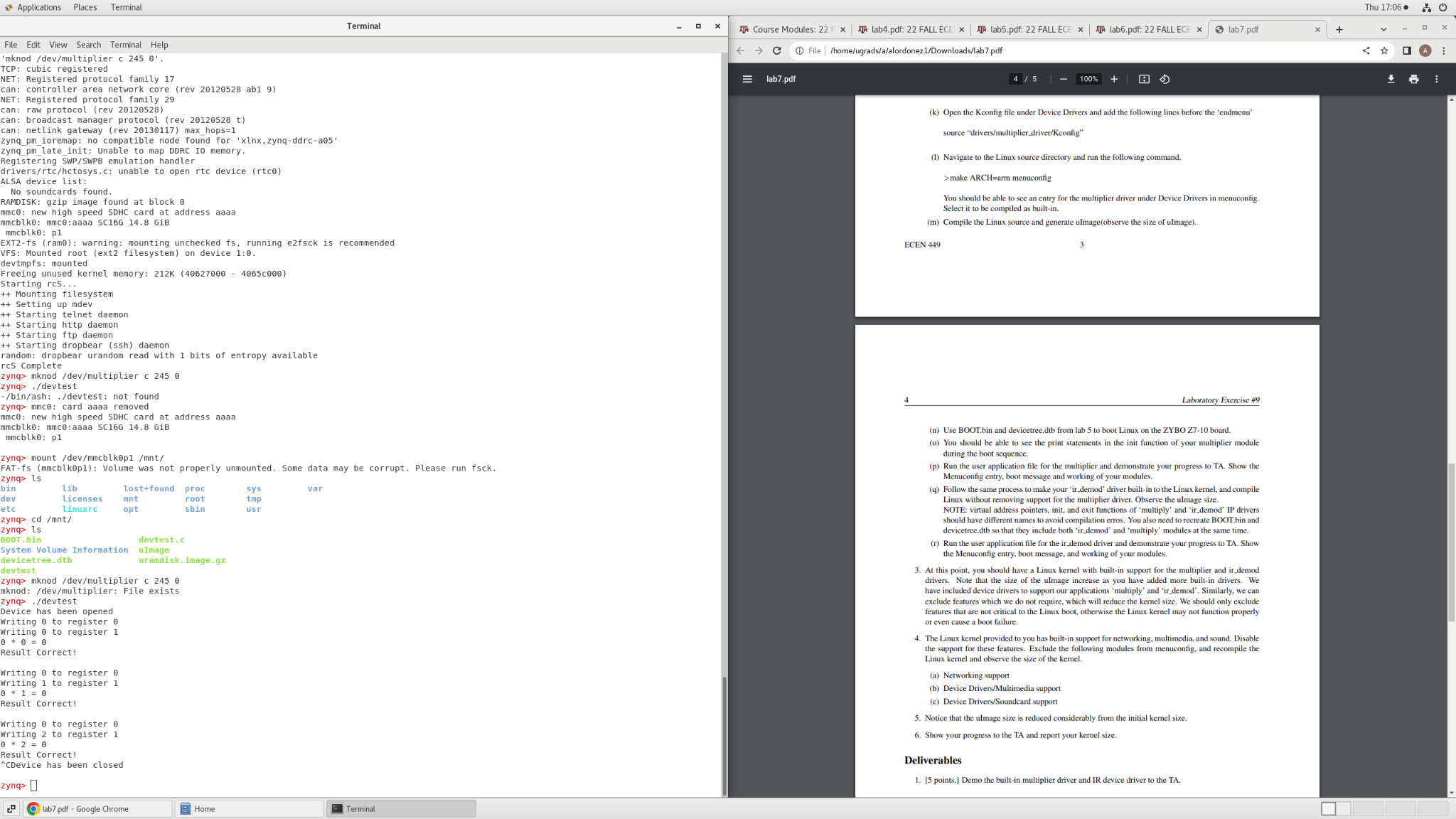
*What are the advantages and disadvantages of loadable kernel modules and built-in modules?*

Loadable kernels allow for a reduced kernel size and the lack of need to recompile the entire module, while still providing all the functionality required for a task. Some disadvantages could be found in the time spent configuring and maintaining the module to ensure things are properly enabled, disabled, or present before compilation. Built-in modules have the advantage of being ready to use without much configuration, and can be booted as is. The configuration is very minimal, especially if we just want to slim down the number of drivers being used in the kernel. A disadvantage would be the time it takes to recompile the entire kernel for every addition or modification, and we have a much bigger kernel size whenever we add more built-in modules.

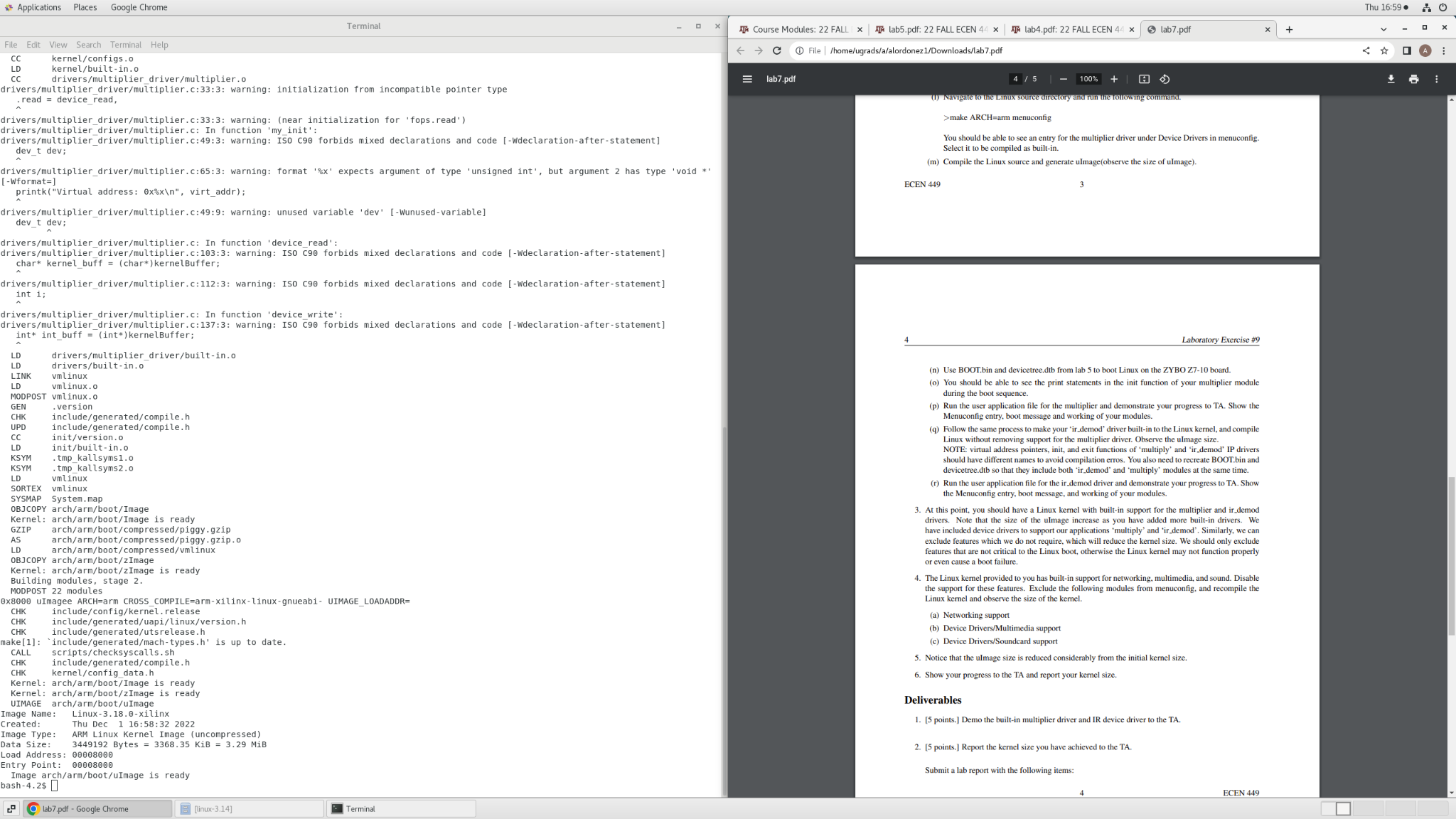
**VI. Appendix**

****

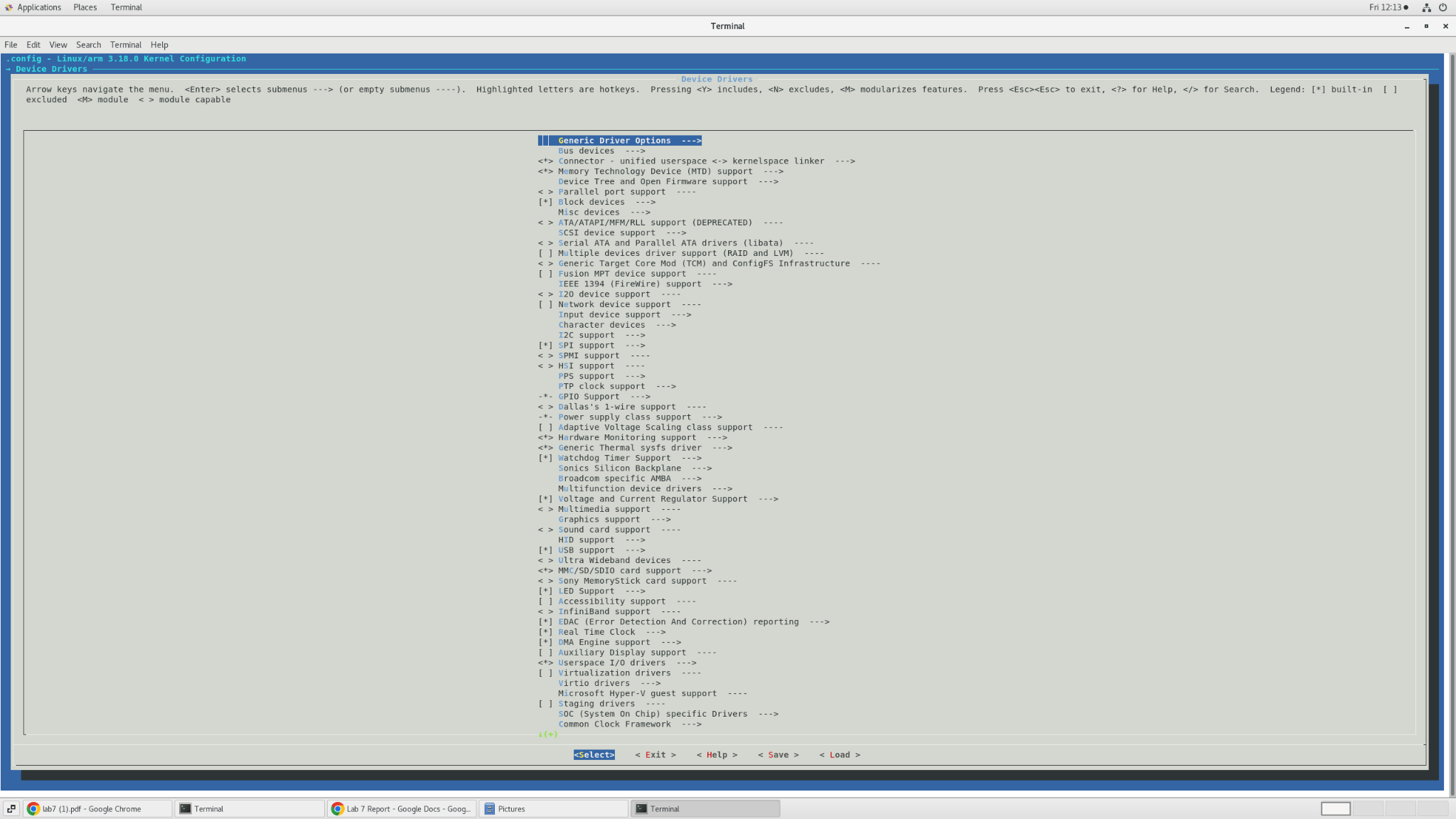
***Figure 1: Multiplier driver accessible and enabled in menuconfig***

￼￼

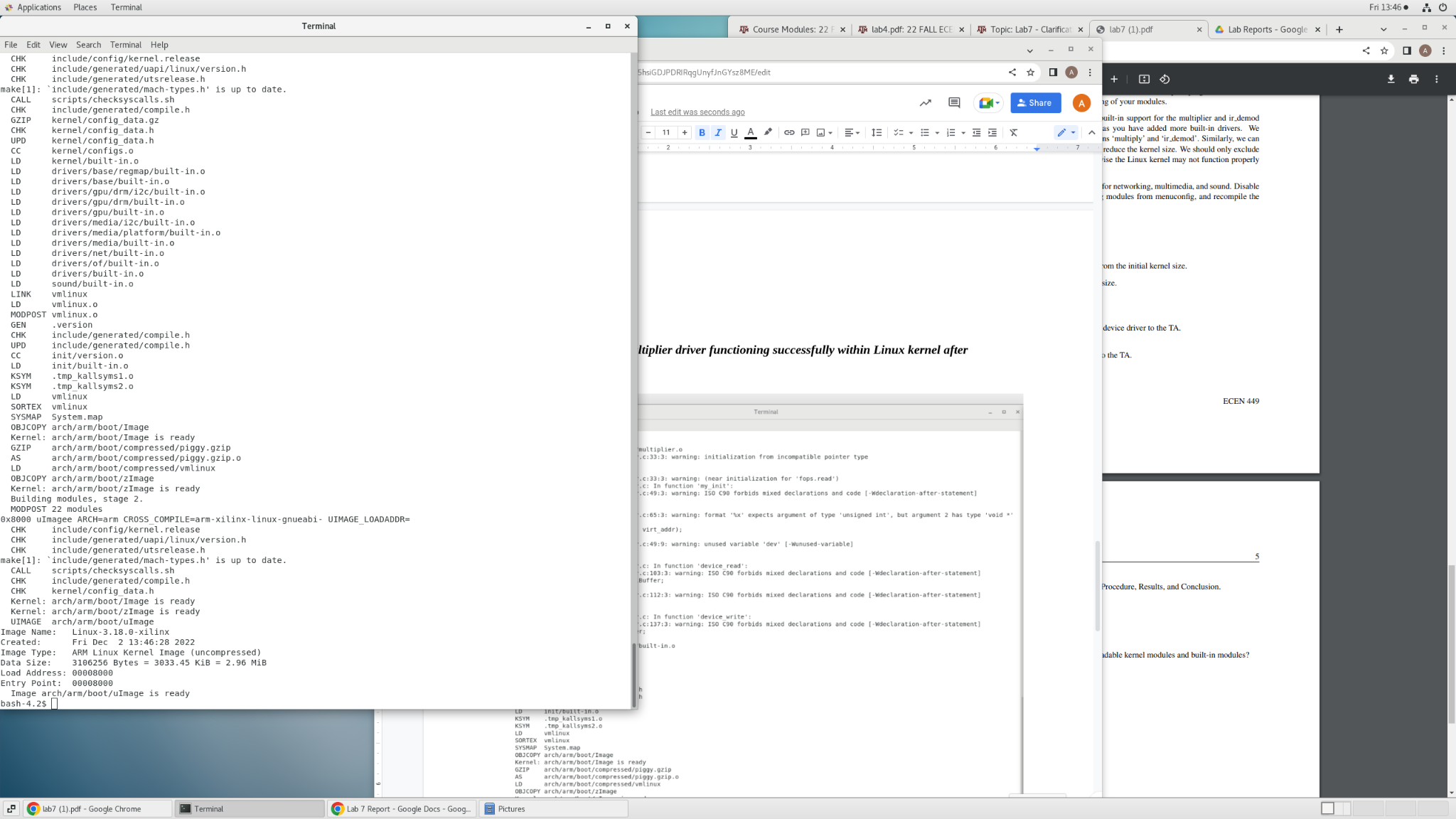
***￼Figure 2: Output of multiplier dr￼iver functioning successfully within Linux kernel after configuration***

******

***Figure 3: Creation of uImage with initial kernel size of 3.29 MiB***

******

***Figure 4: Network device support, sound card support, and multimedia support disabled in menuconfig***

******

***Figure 5: Second creation of uImage with new kernel size of 2.96 MiB***