ECEN 749- Microprocessor System Design

Section 603

TA: Mr. Kunal Bharathi

LAB 9

Linux Kernel: Built-in Modules

Date of Performance: 11/14/2019

Student Name: Dhiraj Dinesh Kudva

UIN: 829009538

**Introduction**

The purpose of this lab to understand the difference between built-in and loadable kernel modules. It also focusses on loading multiple built-in modules and disabling other drivers that are not required thus reducing the kernel image.

**Procedure:**

The device drivers created in the earlier labs where loadable kernel modules and required command to run them after boot up. In this lab, we are using the same device drivers but converting them into bootable kernel modules.

1. Copy the Linux Kernel source code in the LAB 9 directory.
2. Linux supports several architecture and certain features are only for particular architectures. Information regarding this can be found out in Kconfig file.
3. Run the following command to configure the Linux Kernel.

Make menuconfig

1. <\*> indicates that particular driver is built in and < > indicates that driver is loadable one.
2. In order to see differences in the device driver configuration, blackfin driver can be run using the below command

make ARCH=blackfin menuconfig

In this it can be observed that ALSA for SoC audio support is present, as it is dependent on Blackfin architecture. The same was not observed earlier as it is not available for Arm architecture

1. Create a multiplier\_driver folder in the drivers directory of linux source code and copy the source code of the multiplier driver file.
2. Create a makefile in the same directory and copy the below lines:

obj-$(CONFIG\_MULTIPLIER\_DRIVER) += multiplier.o

1. Create a Kconfig file in the same directory and write the below lines

Config MULTIPLIER\_DRIVER

tristate “multiplier\_driver”

depends on ARM

default y if ARM

help

refer to ECEN449@TAMU

1. In the Kconfig file, every line starts with a keyword and can be followed by multiple arguments.
2. Edit the makefile in the drivers directory of the linux source code by adding the following lines

#ECEN 449

obj-$(CONFIG\_MULTIPLIER\_DRIVER)+=multiplier\_driver/

1. Open the Kconfig file in the same directory and add the following lines before the ‘endmenu’

source “drivers/multiplier\_driver/Kconfig”

1. Run the following command

make ARCH=arm menuconfig.

This will create an entry for the multiplier device driver in menuconfig and select it to be compiled as built-in.

1. Check the size of uImage. [in my case it was 3.4 MB]
2. Use BOOT.bin and devicetree.dtb from lab5 to boot linux on the ZYBO Z7-10 board.
3. Now load the SD card on the Zybo board and boot the linux.
4. The multiplier driver would also boot along with the linux. Run the application file to get the outputs.
5. Similarly repeat steps 6 to 15 for ir\_demod driver. This will recreate the uImage and the size of uImage in this case would be 3.5 MB.
6. Remove some of the following drivers and again check the size of uImage.

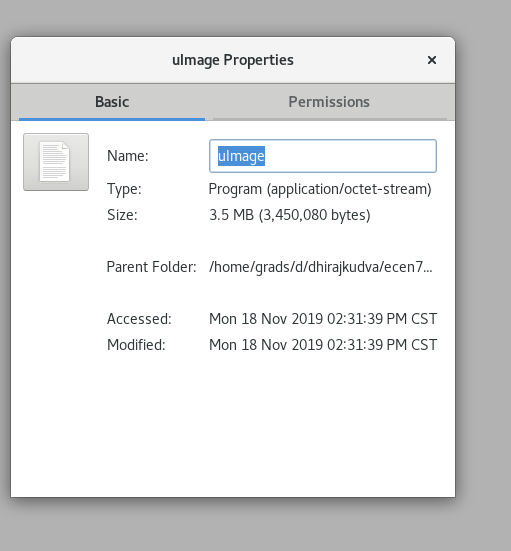
(a) Networking support

(b) Device Drivers/Multimedia support

(c) Device Drivers/Soundcard support

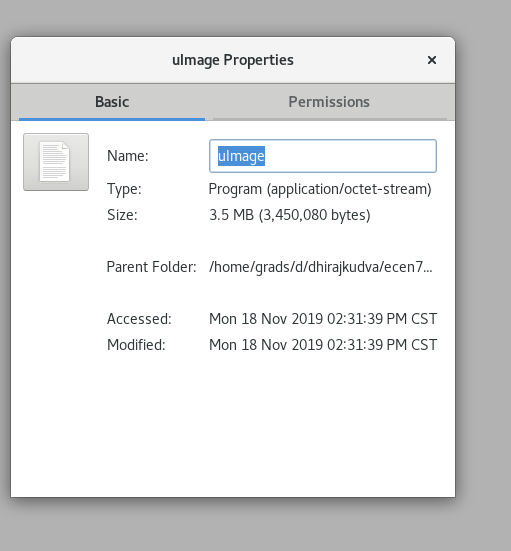
19. In this case, the size of the uImage is 2.5 MB.

**Output:**

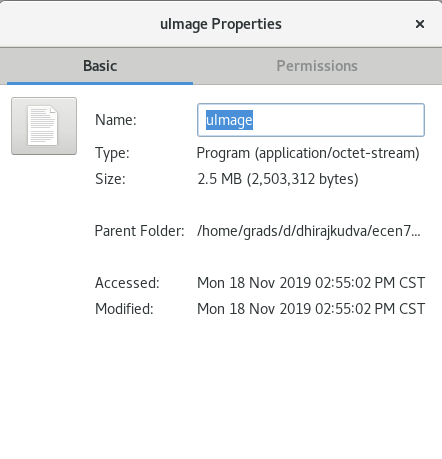
**With single driver**:



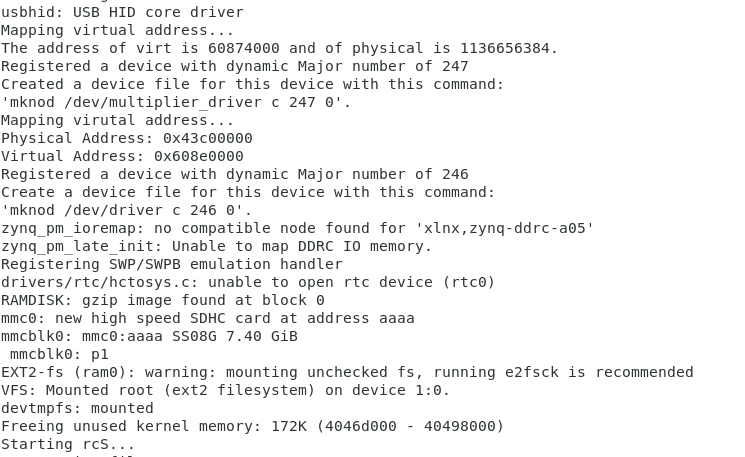
**After loading both the drivers:**



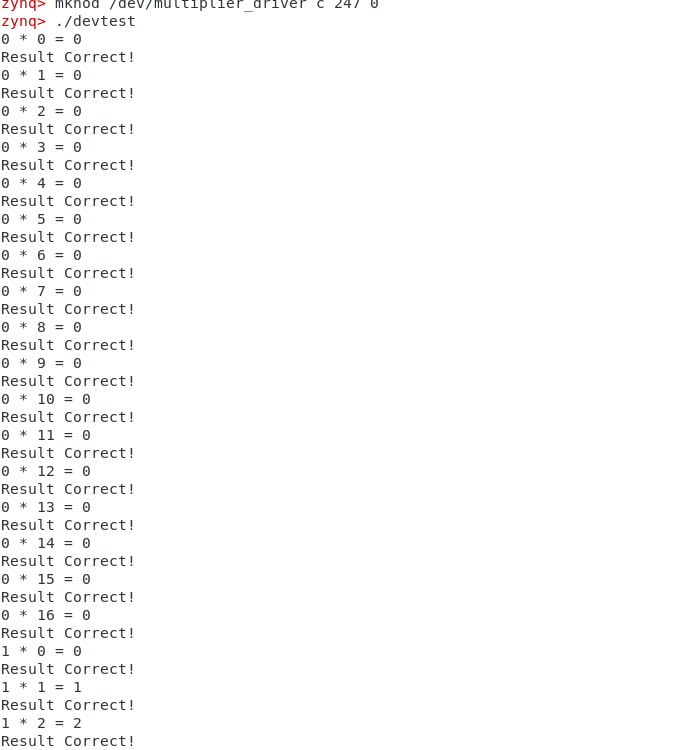
**After removing some of the built-in drivers:**



**Output on the console during boot-up**



**Output of the driver:**



**Results:**

As the new boot files are loaded onto ZYBO, the booted kernel shows that the device is registered. In our case, as two device drivers were made built-in, two device registration messages along with their major number appears. Based on the BOOT.bin file and application file, the device driver which needs to be run can be chosen. The output of multiplier driver appears on the console as the application is run. The enabling and disabling built-in feature for other device drivers can significantly increase or decrease the side of image file.

Size of the kernel image when a single driver is built-in: 3.4 Mb

Size of the kernel image when two device drivers are made built-in: 3.5 Mb

Size of the kernel image on removing networking, sound and multimedia support: 2.5 Mb.

**Conclusion:**

In this lab, we successfully created two built-in modules that booted along with the linux kernel. This helped in understanding the difference between built-in and loadable device drivers. This lab also explained the effect on the size of the image kernel.

**Question:**

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

ANS: In built-in kernel modules, kernel automatically inserts device into the kernel when kernel is booted up. In kernel modules, the driver is not automatically loaded. It is to be loaded by the user using ‘insmod’ command.

The advantage of loadable kernel module is that new modules can be added at any time without booting the OS. This can be done by loading the .ko file of the device driver on the running OS. Modification is easy and also results in smaller size of kernel image.

The disadvantage is that every time the OS is booted, the device drivers are deleted and must be reloaded again. In addition to this, the fragmentation penalty is a major disadvantage of loadable modules in the kernel. This means that every time a new kernel module code is inserted, the kernel becomes fragmented. This leads to a performance degradation.

The advantage of built-in modules is there is no need to re-load the device drivers once the OS is booted, as they won’t be lost when the OS reboots. Also, they are not needed to be loaded everytime the modules are to be used.

The disadvantage is that modification of a particular module is difficult as the entire kernel must be compiled again. Also, the kernel image size increases whenever a module is made built-in.

Therefore, the device drivers/modules that are to be used rarely and of big size, are preferable to be kept as loadable modules.