Embedded OS Internals - Formal Project Report

Aritra Kumar Lahiri
Department of Computer Science, MS
CSE 530 - Embedded OS Internals
Dr. Alan Skousen

Introduction

The Embedded OS Internals semester long course projects has provided a outline and detailed concepts regarding building the in depth concepts in Linux based environments and helped in implementing the kernel and its various functionalities. The entire project work is based on the kernel development concepts and it uses a Beagle-Board as a cross debugging environment to implement the projects. The ultimate objective of the project work has led us to develop our own individual Linux device driver that can provide some services. Since the entire project work was an individual one, hence there were many challenges that needed to be faced and also subsequently carry extensive research on them so that they can be finally resolved. In this formal project report I will touch upon each and individual segments of the semester long project and put forward my learning experiences through each of them.

This project work has made me clearly understand the basic concepts about a kernel and improved my thinking capacity and also made me realize the importance of research in solving the problems.

Project 1a:

The first project was basically about installing the virtual environment in our system for hosting the Linux. For implementing the project I have used the VMware Player 14.0.0.1 as the virtual machine for hosting the Ubuntu Linux 32 bit in my system. There was not any major issues regarding this project. It was about checking configuration of your own system and making necessary modifications while installing the software.

Project 1b:

The second project was about making the SD card bootable for proper use with the beagle board. The SD card is at first part needed to be partitioned and formatted and loaded with necessary instructions that were provided. However there was one issue regarding the calculation of the number of cylinders which was calculated to be 8 per head. For this we had to calculate and put in necessary formula to calculate the final number of cylinders. Again there was an issue regarding the partitioning of the second cylinder where we have to depreciate the number of sectors value to number of cylinders value using -u command. These were the experiences while working on project 1b.

Project 1c:

This project mainly focuses on the serial capture of the Beagle board session. We need to change the Display mode to DVI mode to interact with the desktop of the Beagle board. We then download the VNC viewer to connect to the Beagle board. Here we had to be careful while controlling the Realterm window with necessary modifications. Otherwise the overall experience was fine and we could get the boot term session captured in a text file also while completing this project. This project was the base project for rest of the kernel cross debugging mode parts that we would undergo in the rest of the projects.

Project 1d:

This project tested the skills to develop properly the kernel in remote debugging mode. For this we needed to install the SDK toolchain at first. Then we tried to compile a sample program to check whether the compilation works properly or not using the given commands. However I faced an issue while compiling the program as somehow the compilation code involving the linux-gnueabi command did not work properly. Hence I had to change some of the configuration then the process

ran smoothly. After compilation we used the kernel in remote debugging mode for the first time in the project. This was a good experience to view the program in the GDB console.

Project 1e:

This was the most challenging of all the projects. Here I have to build the kernel for the use in Beagle board. The steps using the bitbake command for building the kernel did not work as mentioned as there was a memory issue in the environment for building. Hence we had to increase the memory for it. Also specific modifications had to be performed regarding the patches installation where I got error previously. After building the kernel we had to configure the kernel in cross debugging mode. This project took a lot of time to complete. There were several bugs and errors in the beginning but we had to research our way out in the process to complete the project. But since the kernel was build properly the rest of the project parts could be done without a hassle .

Project 2a:

In this project we had to add a character driver module to the kernel. To do this project I had to go through the concepts of the book Linux Device Drivers to analyze and learn the basic working principle of implementation of the character drivers. We used a sample code from the book and used it as a module to showcase the loading and unloading of the modules. The driver application program was also built alongside it to test the exercising of the modules of the device driver.

Project 2b:

In this project we had to add the proc file system to the character driver module. For doing the project I had to read about the proc file and its utilities. However after adding the proc files it was compiled successfully, but there was an issue while debugging the module in cross debugging mode. While setting the watch point on the counter value, at first it was seen that the counter value was not properly increasing after the kgdb wait was applied. However after few times after entering the data address, bss address and the text address properly and the list command the issue was solved and we could print the value of each access and the increased counter value after each /proc entry.
This was also a challenging project in the initial but I was able to do it after researching and being patient on the work.

Project 2c:

In this project we had to add the storage capacity to the driver. For this I had to study and analyze which process among the kmalloc(), vmalloc(), kmem_cache_alloc() are suitable in Linux environment for adding the storage capacity to the driver module. Based on the differences I chose kmem_cache_alloc() to do the adding of storage capacity. There are two applications of the driver module where the write application writes the total content size into the buffer. The read application on the other hand reads the content of the buffer and then frees the buffer on its one invocation. In this way the write and read operations work antagonistically to maintain the contents of the buffer in the character driver module.

Project 2d:

This project is basically about synchronizing the driver application that we have built so far. To do this project I had to analyze the differences between the semaphore and spinlock to choose which one among them is the best one to complete the functionality asked in the question. Based on their differences I had chosen the semaphore as the best option to do the synchronization. Here thus we can implement multiple read and write operations concurrently in a synchronous fashion to do the project. This was also a seemingly challenging one because of a certain bug in the code, however upon deep reviewing I was able sort out the problem and complete the project.

Project 3a:

This project is about giving the basic synopsis of what we are going to implement as a part of the final project. I submitted a proposal on the device driver module implementation with the functionalities of the previous modules and also providing certain string manipulations services such as reversal, printing the size, calculating the number of digits, upper cases or lower cases.

Project 3b :

This was the final project submission of the entire course project. It was about implementing your own device driver to provide some services. I used the concepts that I studied while doing the parts of the project 2 here and also along with it implemented certain string manipulation functions such as reversal of the characters of the string including the whitespaces and also printing the size of the string and its contents among other functions.

Summary:

Thus the above mentioned points describes the entire course project in part by part manner and also highlights the experiences faced while doing them. This was a great learning curve for me to be able to dig in and finish through the entire project successfully. Some of the parts were challenging but it was a satisfying experience to complete the project. To summarize in the end after giving the overview of the projects I want to add that my basic understanding of Linux based kernel have been thoroughly enriched and will help me immensely in my course work ahead.

Acknowledgement:

I want to acknowledge Professor Alan Skousen immensely for designing this course in a very effective way. This has helped not only in developing our application skills but also made us learn more by researching and digging deep into a particular topic. I am also thankful to all my classmates who have time and again helped me by providing valuable inputs while doing the projects in the laboratory.

Bibliography:

[1] Corbet Jonathan, Rubini Alessandro, Kroah-Hartman Greg, *Linux Device Drivers*, 3rd edition, USA, O'Reilly Media Inc, 2005

[2] Bovet Daniel P., Cesati Marco, *Understanding the Linux Kernel*, 3rd edition, USA, O'Reilly Media Inc, 2005.

[3] Hallinan Christopher, *Embedded Linux Primer*, Second Edition, Boston MA, Prentice Hall Professional Technical Reference, 2010.