ESE 333: Real Time Operating System

Linux and Linked Lists
Project 1

1 Introduction

In this assignment you are to install VirtualBox, import a provided OS, make and load a provided kernel module, and write some linked lists functions in C.

2 Linux

This semester you will work in Ubuntu Linux though a VirtualBox environment for the course projects. VirtualBox is a tool that allows users to run additional operating systems within their local operating system. You can download VirtualBox for Windows, Linux, or MacOS at

- VirtualBox Download

Download the .ova file here: .ova file. This 4GB file will take a while to download and must be imported into VirtualBox. You will need a machine with at least 40GB of available disk space to use the provided operating system.

2.1 Importing .ova file

Once you have VirtualBox installed, select the "Tools" item on the bar on the left. An option called Import should appear in the top bar. Select this and navigate to the location of the provided .ova file. Don't change any of the settings and select import. This will install an Ubuntu 18.4 operating system running kernel version 5.3.0-26-generic. The username and password are: "ese333".

2.2 Basic Linux commands

Before you start writing C, make sure you are comfortable with some basic Linux terminal commands. You can open a Linux terminal by right clicking and selecting "Open Terminal". Familiarize yourself with the following list of basic commands. You can find an explanation of these commands here.

- ls
- cd
- echo
- cat
- cp / mv
- pwd
- rm
- mkdir / rmdir
- man
- sudo
- uname
- chmod

3 Hello World Kernel Module

As you will learn this semester, programs can be written to run in user space or in kernel space. User space programs take advantage of the operating system abstraction and are therefore simpler. Kernel modules are designed to be drivers not applications. Drivers need to be initialized upon install and free resources upon removal/shutdown.

3.1 Writing Kernel Code

Kernel code does not have *printf* or any of the standard c functions (they are user space packages) but it has *printk* which can be used to print information about the kernel. *printk* prints messages to the kernel log that can be obtained from the userspace using simple command "dmesg". The format of *printk* is as follows:

```
Printk(LEVEL ,''message'',..);
```

Level is type of message this is (e.g. information, debugging, error, critical, etc.). In your linux source code, please refer to file *include/linux/kern_levels.h* to understand and know which levels you can use in *printk*. Then please refer to kernel.h file in *include/linux/kernel.h* and understand how that ties to *kern_level.h* and *printk*.

You have been provided with a program *kernel_demo.c* as well as the associated make file. It can be found at: */home/ese*333/*ESE*333/*Proj*1/*Kernel_Demo* on the VirtualBox image that has been provided or can be downloaded from brightspace.You will need to open the program, change the name of the creater, and delete the comment on the *printk* command.

What this code does is, upon installation, calls function *hello_world*, and upon exit calls *goodbye_world*. It would also be in your best interest to look at the make file for this program.

That is it! All you must do now is type command "make" and the module is built. Try installing the kernel object by typing "sudo insmod kernel_demo.ko" and check using command lsmod to confirm your custom module is installed. Then check dmesg to ensure your print message occurred. Then try removing it using "sudo rmmod kernel_demo" and check dmesg again.

3.2 Kernel Module Project Deliverable:

Modify the hello statement to include your name and the goodbye to include your ID number and practice loading and removing the kernel object. Time will be set aside in class where you will show the dmesg output of this program to me.

4 Linked List and C practice

You will make use of linked lists in many of the projects this semester and linked lists serve as a nice way to introduce the C programming language. You have been provided a program that constructs a sorted linked list. You will implement the insert and delete methods for this program.

4.1 Linked Lists

Linked lists are a data structure where all entries contain some data as well as a pointer to the next entry. This means that adding and removing elements from this data structure only require you to modify pointers and allocating or freeing memory. Traditionally a linked list is defined by two pointers, one that point to the head (or first element) for easy traversal and searching, and one that points to the tail (or last element) for easy insertion.

The linked list in this project will only include a head because it is a sorted list. This means that to insert or delete an element, you will need to traverse the list until you find the appropriate location of the new element.

4.2 Memory management and struct in C

The C programming language does not have objects like C++/C#/Java. To accomplish the same idea as objects (composite data types) you must use the keyword struct. A struct is simply a block of contiguous memory that can be made to contain primitive data types as well as pointers to other structs. The node struct in the sample program includes a pointer to a second node as well as an integer called data. A pointer is the address of a location in memory that contains data like a struct or a function. If you are new to the C programming language, study the linked tutorial as it covers much of what is required of this project. C tutorial

Look at the main of the sample program to see how you can access and assign these data members with the $->$ operator. You will also notice the keyword "malloc" which stands for memory allocate. When you want to initialize a new struct you need to allocate memory for this struct. This will create a space in memory for the struct that can then be assigned to whatever data or pointer value you wish. When you write your delete method you will also need to use the "free" command. free is the opposite of malloc, it will free (deallocate) the memory that the structure or pointer is using. This is critically important in kernel programs and userspace programs that allocate large amounts of data. Examples of malloc and free can be found in the sample program.

One common mistake new C programmers make when using structs and malloc is to incorrectly assume that all pointers need to have memory allocated. A pointer will have the necessary memory to store the address being pointed to allocated when it is defined, malloc is only used when the data being pointed to is both not a primitive and not previously allocated.

You have been provided with the program *linkedList.c* as well as the associated make file. It can be found at: */home/ese333/ESE333/Proj1/linkedList_Demo* on the provided VirtualBox image or can be downloaded from brightspace. This program defines the struct

you will need and constructs a sorted linked list with the even numbers between 0 and 10. Then it calls insert and delete on various numbers to test your methods. Finally it will print out the current contents of the list. You can build this program by typing "make" and execute the program by typing "./linkedList".

4.3 Linked List Project Deliverable:

Implement insert and delete functions for this sorted linked list within the provided sample code. Insert will take in a pointer to the list and the value of the new entry and return the updated list. Delete will take in a pointer to the list and the value of the list to be deleted and returns the list, if the value to be deleted is not in the list, then the list is not changed. Submit your functional linkedList.c program on Brightspace.

5 Useful References

- C tutorial
- VirtualBox Download
- Hello World Kernel Module
- kern_levels.h source code
- kernel.h source code