M1522.000800 System Programming
Fall 2018

System Programming Kernel Lab Report

Lee Young Jae

2016-11988

1. Kernel Lab

The goal of the kernel lab is to learn basic kernel module programming and understand the difference between kernel-level programming and user-level programming. To put the new feature into the kernel, we originally need to recompile the kernel. In my experience with installing Gentoo, it takes about 2 hours to compile, which is very annoying and risky. Instead, we can safely add new module with `debugfs` with very short time, since the modules we added disappear when we reboot the computer. The goal of this lab is adding two modules `ptree`, and `paddr`. What exactly each module does is introduced in the next chapter.

2. Implementation

The kernel developers have to follow the convention for Linux Kernel Module. A basic frame for Linux Kernel Module is `init_module` and `exit_module`. The formal is called when the kernel module is inserted to system, and the latter is called when the kernel module is removed from system. The two functions are enrolled to the kernel using module init and module exit functions. Here is a basic structure for kernel module programming in `debugfs`:

```
static ssize_t operation(struct file *fp,
                         const char __user *user_buffer,
                         size_t length,
                         loff_t *position)
{
    // Operation Details
}

static const struct file_operations dbfs_fops = {
    .operation = operation,
};

static int __init dbfs_module_init(void)
{
    // Some Codes
}

static void __exit dbfs_module_exit(void)
{
    // Some Codes
}
```

In each assignments, `ptree` and `paddr`, it has skeleton C code and build script. My task is implement to complete each codes. Fortunately, I did not have to fix the Makefile.

2.1. ptree

The purpose of this Assignment is tracing process from the leaf to `init` process and logging it using `debugfs`. ssibal

```
printf("Hello, World!");
```

2.2. paddr

3. Conclusion