Practice Lab 1b - Linux Kernel Modules and Process Representation in Linux

Operating Systems CS 3SH3 Term 2, Winter 2018
Dr. Neerja Mhaskar

Practice Labs are not mandatory, they are primarily designed to help you with your Lab tests and assignments.

Lab Format: Practice labs will be posted a day before or on the day of the lab on the course website. You can choose to solve it beforehand and come in with your solutions and check the correctness of your solution with your TA.

The TAs will also be available to answer any questions you might have on your assignments.

Outline

In this lab you will learn the following:

1. How to create a kernel module and loading/removing them from the Linux kernel.
2. How processes are represented in Linux and how to access members of the idle/swapper process.

This lab should be completed using the Linux virtual machine that you installed in Practice Lab 1a. Although you may use an editor to write these C programs, you will have to use the *terminal* application (on the virtual machine) to compile the programs, and you will have to enter commands on the command line to manage the modules in the kernel. As you'll discover, the advantage of developing kernel modules is that it is a relatively easy method of interacting with the kernel, thus allowing you to write programs that directly invoke kernel functions. It is important for you to keep in mind that you are indeed writing *kernel code* that directly interacts with the kernel. That normally means that any errors in the code could crash the system! However, since you will be using a virtual machine, any failures will at worst only require rebooting the system.

Part-1 Linux Kernel Modules

Follow the steps under Linux Kernel Module on page 96 of the textbook for

a. Creating Kernel Modules, and
b. Loading and removing kernel modules

If you followed the above steps, you would have written a C program called simple.c.

Part – 2: Process Representation in Linux

You can use the file (simple.c) you created under Part-1 of this lab to complete Part-2.

Processes in Linux are represented by the `task_struct` data structure as shown in Chapter 3 lecture notes. This data structure is defined in the sched.h header file. Take a look at the definition of this data structure and study the following members of this structure:

`pid, state, flag, rt_priority` (runtime priority), `process` (process policy) and `tgid` (Task group id).

In this part of the lab, you will write a Linux kernel module which outputs the values of the above listed members to the kernel log buffer for the `init` task (also called the swapper/idle process) in a Linux system, when the module is loaded into the kernel.

Print the output of the `dmesg` command to a file and show it to your TA.

Note:

1. The `task_struct` data structure for the Linux swapper (idle process) is named `init_task` and it has a `pid = 0.` It is a task scheduled to run when no other process exists to run on the system.
2. If you are using the `simple.c` file, you need to only write a procedure `print_init_PCB()` that prints the required members of init_task. Then call this function in `simple_init().`

SAMPLE OUTPUT

```
Loading Module
[ 2300.008729] init_task pid:0
[ 2300.008733] init_task state:0
[ 2300.008736] init_task flags:2097152
[ 2300.008740] init_task runtime priority:0
[ 2300.008743] init_task process policy:0
[ 2300.008745] init_task tgid:0
```