CS342 Spring 2015 - Project 4
Writing a Linux Kernel Module and Accessing VM Information

```
Assigned: April 10, 2015
Due date: April 26, 2015
```

In this project, you will develop a Linux kernel module and in this way touch to the Linux kernel. You can do the project in Linux installed on bare hardware or on a virtual machine. At the end, you will write a report and you will be called for a demo. Below are the project steps. We suggest you to do the project on a 32-bit Linux OS. If you wish you can also do it on a 64-bit Linux OS. We suggest you to do the project in a machine with Intel x86 architecture.

1. Learning how to build a Linux kernel. [If you wish you can skip this step 1 if you succeed doing step 2 without doing this step 1]. First learn how to compile (build) and run a new Linux kernel, so that you can get prepared to write a kernel module. Learn from Internet how to build and run a new kernel. Download source code of Linux kernel, build it (this may take a while – one hour or so – the first time you do it) and run it. You can do it on a virtual machine if you wish. If you are doing it directly on your machine, make sure you backup all your data first, so that if you mess up the file system and partitions on the disk, you can recover your data. Explain in your report briefly how you built and run your kernel. Note that this part will be quite time consuming, but you will do it only once (until you get your new kernel running).

2. Learning how to write a module and develop a simple Hello World module. In this step you will learn how to develop and run (load-insert) a new kernel module. Compiling a module and loading/running is very easy and fast (just a few seconds) after you have the right development environment set up. There is documentation available on the web about Linux kernel module programming. Search for "Linux kernel module programming". Below are two good references to start with. They can be reached from the course website. Read this documentation and do some simple exercises.

- The Linux Kernel Module Programming Guide,
http://www.tldp.org/LDP/lkmpg/2.6/html/index.html

- Linux Device Drivers, Third Edition, http://lwn.net/Kernel/LDD3/, (especially the
Chapter 2: Building and Running Modules).

3. Develop the module. In this step, you will implement a Linux kernel module to get information about processes created in the system, about the memory usage of a particular process, and about the open files of a proces. Your module will retrieve the required information from the related kernel data structures (PCBs, virtual memory regions of a process, page table of a process, and open file structures). Your module will be a kernel code that is loadable and unloadable while a kernel is running. It will be loaded using the *insmod* command. The module will take one argument, a process identifier (an integer value), while loading it dynamically using insmod. When loaded, the module becomes part of the kernel and runs in kernel mode and space. Below are the things that your module will do.

- Print the process tree in a nice format you will decide. Each node will give information about a process (its process id), its parent and its children. You can also write the name of the process. Your module will need to traverse the process list (PCB list). There is a "current" variable in Linux kernel that is pointing to the PCB (of type tast_struct *) of the currently running (scheduled) process. Starting from “current”, you can traverse the list of PCBs. The PCBs in ready and running states are linked together (double linked list).

The printing will be done using the printk() function and output will go to a kernel log file that can be examined later by using commands like dmesg, more, cat, tail, etc.

- Print virtual memory (VM) layout information of the process whose pid is specified at command line while your module was being inserted. Information about the VM regions that are used by the process will be printed out. For each VM region, you can print out, for example, the start virtual address, the end virtual address, and the size of the region.

You need to traverse the PCB list to find the PCB of the desired process. After finding the desired PCB, you will need to reach memory management related information from the PCB. Check, for example, the mm field of the PCB and the definition of the related structure. Also compute the total virtual memory used by the process.

The following is a very good book on Linux Virtual Memory Manager:
https://www.kernel.org/doc/gorman/pdf/understand.pdf

You can verify your results using the output of such tools/commands: top; ps aux; cat /proc/pid/maps; cat /proc/meminfo; cat /proc/vmstat; cat /proc/zoneinfo.

You can also verify your results with the output you can obtain from the /proc file system. Go into directory /proc. There you will see folders with integer names. Those integers are process ids. Change into the folder with name 'pid'. Type 'more' or 'cat'. There you will see some files. They are actually virtual/special files whose content is not sitting on disk. The content for these files is obtained from memory resident kernel structures. Type 'cat maps', for example to see the virtual memory regions of a process. Hence the /proc file system is a special file system corresponding to the kernel state. If you want to learn information about the kernel state, you can change into this /proc directory, traverse and obtain kernel state information. It is the interface of the kernel state to the users to learn about the values of some kernel variables and structures. The name /proc file system comes from 'process file system'.

- Print the top-level page table content of the process. Parse each entry and write the information nicely. 32-bit Intel machines are using 2 level page tables. 64-bit Intel machines are using 3 level page tables. This part requires architecture knowledge (structure of the page table and page table entry the machine is using, etc.).

- Print open files information. You will reach to the information about the open files of a process from its PCB. Access that information and print out some information about each file opened by the process. You can print the handle of the file (file descriptor), name of the file, size of the file, first 3 block numbers used by the file, etc.

4. Report and Submission. You will also write a report at the end. You will write in your report how you implemented your module, how you tested it, which information you are printing out, etc. Sample outputs will be included in the report. Include your module code as well. You will also upload your project (report, module, etc.) to Moodle. Make sure you include the names of the group members in your report . One submission per group is enough.

You will do a demo of your module. You will bring your machine. It is also possible that we can do oral or written exams. Make sure each group member is working and learning. We can also have questions from the project in the exams.

Other Issues:

- Check the Clarifications page for further requirements, information, explanation, and clarification.