CSCI 5573: Advanced Operating Systems
Fall 2018

Programming Assignment One

Due Date and Time: 11:55 PM, Monday, October 08, 2018

The goal of this programming assignment is to get familiar with the proc filesystem, develop some utilities using the proc filesystem, and hack the virtual memory address space of a process.

Please work with a virtual machine. You will need super user privileges to do some parts of this assignment.

You may work in teams of size two students.

1. Monitoring the kernel state through /proc

   Write a program to report the behavior of the Linux kernel. Your program should run in two different versions.

   The default version should print the following values on stdout:
   - Processor type
   - Kernel version
   - The amount of memory configured into this computer
   - Amount of time since the system was last booted

   A second version of the program should run continuously and print lists of the following dynamic values (each value in the lists is the average over a specified interval):
   - The percentage of time the processor(s) spend in user mode, system mode, and the percentage of time the processor(s) are idle
   - The amount and percentage of available (or free) memory
   - The rate (number of sectors per second) of disk read/write in the system
   - The rate (number per second) of context switches in the kernel
   - The rate (number per second) of process creations in the system

   If your program (compiled executable) is called proc_parse, running it without any parameter should print out information required for the first version. Running it with two parameters "proc_parse <read_rate> <printout_rate>" should print out information required for the second version. read_raterepresents the time interval between two consecutive reads on the /proc file system. printout_rate indicates the time interval over which the average values should be calculated. Both read_rate and printout_rate are in seconds. For instance, proc_parse 2 60 should read kernel data structures once every two seconds. It should then print out averaged kernel statistics once a minute (average of 30 samples). The second version of your program doesn't need to terminate.

2. Virtual memory layout of a process

   Write a program that includes multiple calls to *malloc( )* as well as several function calls. Using the proc filesystem, print out the virtual memory address layout of this process during different execution phases – before calling *malloc( )*, after calling *malloc( )*, before calling a function, while a function has been invoked, after the function returns, etc. The layout should clearly show the address ranges of the executable code, heap and stack. Answer the following questions:
   - In what directions do the stack and the heap grow? Explain your answer by using the virtual memory layouts at different execution phases.

CSCI 5573: Advanced Operating Systems
Fall 2018

- Your executable code will be “divided” into three different regions with different permissions. What are all these regions?
- Your allocated memory (from *malloc( )*) will not start at the very beginning of the heap. Explain why this is so.
- The program break is the address of the first location beyond the current end of the data region of the program in the virual memory. *sbrk()* with an increment of 0 can be used to find the current location of the program break. In general, that should be the start of the heap. However, you will see that this is NOT the case. The only thing that is true is that the heap is always the next memory region after the executable. If you run your program between the executable and the heap is random. Explain why this is so.

3. Hacking the virtual memory of a process

   Write a program that creates a copy the string “This is my initial string ” and prints this string repeatedly (indefinitely) with an interval of 5 seconds between each print out. Using a scripting language such as Python or bash, write a script that locates the virtual address of the string (first program) using proc filesystem and replaces that string with another string “This is a different string”. Demonstrate your program by first running the first program and then running the script. You must run your script as root.