CS 111: Operating System Principles
Lab 0

A Kernel Seedling 3.0.0

Yadi Cao, Brian Roysar
Derivative document by: Jonathan Eyolfson
September 27, 2023
Due: Oct 06, 2023 @ 11:59 PM PT

In this lab, you'll setup a virtual machine and write your (probably) first kernel module. We'll use VirtualBox as our hypervisor since it supports many different host operating systems, and is friendly to learn. Finally, you'll write a kernel module that adds a file to `/proc/` to expose internal kernel information.

Virtual machine setup. After the setup you'll have a fully functioning Linux virtual machine. You're free to edit your files with whatever you're comfortable with. For example, `vscode`,`emacs` or `vim`. You should only run your code on the virtual machine. Note:If you are using M1 Macbook follow the m1 virtual machine setup guide.

1. Download and install VirtualBox: https://www.virtualbox.org/wiki/Downloads
2. Download our OS image: https://ucla.box.com/s/8fx06w3rwn8au0h04rhc3opho0zux343.
   If your computer is an old model, use the ligthweight version of the vm https://ucla.box.com/s/arvh779n3ggchsekj5nxclyna0iwk2ce
   For the lightweight vm, first login through the terminal, then run startx command to start the desktop environment if you are not comfortable with the command line interface.
3. Import the virtual machine
   (a) *File* → *Import Appliance*
   (b) Choose `vm.ova` from your local file system
   (c) *Next* → *Import*
4. Select *CS 111* from the left panel and click *Start* at the top of the right panel
5. Use `cs111` for both the username and password
6. (Optional) Go to *View* → *Virtual Screen 1* and resize to any resolution you'd like
7. Note: Using the power off option might cause errors in your active files. Use graceful shutdown/reboot options instead. You can also use 'save-state' option.

Your task. You're going to create a `/proc/count` file that shows the current number of running processes (or tasks) running. The process table runs within kernel mode, so to access it you'll need to write a kernel module that runs in kernel mode. For your submission you'll modify `proc_count.c`, and only this file, for the coding part. In the `lab0` directory we should be able to run the following commands:

```
make
sudo insmod proc_count.ko
cat /proc/count
```

The last command should report a single integer representing the number of processes (or tasks) running on the machine. Your final task is to fill in your documentation in the `README.md` for `lab0`.

Tips. The kernel code is well commented, you can use https://elixir.bootlin.com/ for looking up functions and macros (symbols). There's already a skeleton that uses: `MODULE_AUTHOR`, `MODULE_DESCRIPTION`, `MODULE_LICENSE`, `module_init`, `module_exit`, and `pr_info`. You'll probably want to use the following to complete this lab:

```
proc_create_single
proc_remove
for_each_process
seq_printf
```

You can divide this task into small subtasks:

1. Properly create and remove `/proc/count` when your module loads and unloads, respectively
2. Make `/proc/count` return some string when you `cat /proc/count`
3. Make `/proc/count` return a integer with the number of running processes (or tasks) when you `cat /proc/count`

Commands. You'll have to use the following commands for this lab:

Build your module with `make`
Insert your module into the kernel with `sudo insmod proc_count.ko`
Read any information messages printed in the kernel with `sudo dmesg -l info`
Remove your module from the kernel (so you can insert a new one) with `sudo rmmod proc_count`
Sanity check your module information with `modinfo proc_count.ko`

Testing. There are a set of basic test cases given to you. For this lab the provided test cases are likely the ones we'll use for grading. In the future we'll withhold more advancded tests which we'll use for grading. Part of programming is coming up with tests yourself. To run the provided test cases please run the following command in your lab directory:

```
python -m unittest
```

Grading. The breakdown is as follows:

75% code implementation in `proc_count.c`
25% documentation in `README.md`

Submission.

1. All lab submissions will take place on BruinLearn. You will find submission links for all labs under the assignment page.
2. The submission format is a single .tar.gz file. This archive should include all files that were given to you in the skeleton(create a tar.gz file from your lab directory). Do not inclue any executable, pycache directory etc that are not inclued in the skeleton code directory. You should only modify the skeleton code and README.md in this directory. The name of the file should be your student ID with no separators (eg: 4051238888.tar.gz).
3. Any submission that does not follow the submission guideline will receive -10pts.