CS 111: Operating System Principles
Lab 0

A Kernel Seedling 1.0.0

Jon Eyolfson
March 29, 2021
Due: April 9, 2021

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. You'll be using Git to submit your work and save your progress. 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'll also have a shared folder accessible on both the host and guest operating system. You're free to edit your files with whatever you're comfortable with in whichever operating system you wish. You should only run your code on the virtual machine though.

1. Download and Install VirtualBox: `https://www.virtualbox.org/wiki/Downloads`
2. Download our virtual machine: `https://laforge.cs.ucla.edu/cs111/media/cs111/vm.ova`
3. Import the virtual machine
   (a) *File* → *Import Appliance*
   (b) Choose `vm.ova` from your local file system
   (c) *Next* → *Import*
4. Create a folder called *Shared* somewhere on your machine
5. Add the *Shared* folder to the virtual machine
   (a) Right click *CS 111* from the left panel and hit settings
   (b) Click on *Shared Folders* on the left panel
   (c) Right click in the right panel and select *Add Shared Folder*
   (d) For *Folder Path* select *Other...* and select your *Shared* folder on your machine
   (e) Leave the *Folder Name* as *Shared*
   (f) Check *Auto-mount*
   (g) Press *OK*
   (h) Press *OK*
6. Select *CS 111* from the left panel and click *Start* at the top of the right panel
7. Use `cs111` for both the username and password
8. (Optional) Go to *View* → *Virtual Screen 1* and resize to any resolution you'd like

Git setup. Run all these commands in your *Shared* folder on your virtual machine. First, open a terminal by going to *Activities* and selecting the *Terminal* icon on the left. Your *Shared* directory is mounted at `/media/sf_Shared`. If you're unfamiliar with Git, please check out the Pro Git book. For any of the commands, run them in the terminal.

1. Run: `git config --global user.name "Your Full Name"`
2. Run: `git config --global user.email your@email.com`
3. Run: `ssh-keygen -o`
   (a) Press *Enter* for the default location
   (b) Press *Enter* for no passphrase
   (c) Press *Enter* again to confirm
4. Login to the course website
   (a) (Optional) Go to *Activities* and click the *Firefox* icon on the left
5. Click your username in the top right
6. Click *New SSH Key (Text Input)*
7. Add your SSH key
   (a) Run: `cat ~/.ssh/id_rsa.pub`
   (b) Copy and paste the contents into the text box
   (c) (Optional) Give the key a comment (it'll be its name)
   (d) Press submit

8. Run: `cd /media/sf_Shared`
9. Run: `git clone git@laforge.cs.ucla.edu:spring21/username/cs111`
10. Run: `cd cs111`
11. Run: `git remote add upstream git@laforge.cs.ucla.edu:spring21/jon/cs111`

Lab Setup. Ensure you're in the `/media/sf_Shared/cs111` directory. Make sure you have the latest skeleton code from us by running: `git pull upstream main`. You can finally run: `cd lab-00` to begin the lab.

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 `lab-00` 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 `lab-00`.

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
IS_ERR
PTR_ERR
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`

Grading. The breakdown is as follows:

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

Submission. Simply push your code using `git push origin main` (or simply `git push`). For late days will we look at the timestamp on our server. We will never use your commit times as proof of submission, only when you push your code to the course Git server.