CS 423 Operating Systems Design
MP0: The Kernel Development Environment

1 Goals and Overview

- In this MP you will learn download, compile, and install your own kernel.[1]
- You will configure your development environment for upcoming projects
- You will familiarize yourself with the layout of the kernel's code
- The kernel source code will be a helpful reference for upcoming MPs

2 Developmental Setup

Each student will be assigned one VM with two 2.3GHZ CPUs and 4 GB RAM running Ubuntu 16.04. You will use this VM to complete all of the MPs in CS423. You will be able to log in using your university netid and password. We will review in class how to use vSphere (`https://vc.cs.illinois.edu/vsphere-client`) to power on/off your VMs. Please ONLY power on/off your own group's VM.

You will work on the provided Virtual Machine and you will develop kernel modules for the Linux Kernel 4.4.0-109 which is used by the OS in this Virtual Machine. You will have full access and control of your Virtual Machine, you will be able to turn it on, and off using the VMWare Server Console.

Start configuring your Virtual Machine for kernel module development by downloading the Linux Kernel headers and standard development tools:

[1] The instructions in this document are adapted from the guide at `https://wiki.ubuntu.com/KernelTeam/GitKernelBuild`

```
sudo apt-get install git build-essential kernel-package fakeroot libncurses5-dev
libssl-dev ccache libelf-dev
```

Kernel compilation can take quite awhile. Moreover, as you are SSHing into the machine, a network connectivity problem or errant closed window could cancel the build. Therefore it would be wise to learn how to use the `screen` tool:

```
sudo apt-get install screen
```

Now, let's download the kernel source git repository:

```
git clone
git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
```

You can check to see which version of the kernel this is by opening the `Makefile` and reading the `VERSION`, `PATCHLEVEL`, and `SUBLEVEL` values at the top of the file. Running a newer version of the kernel than Ubuntu 16.04 was designed for might work without issue (feel free to try it), but it could also cause problems as significant changes sometime occur within the kernel. To be safe, let's checkout the appropriate version of the kernel source:

```
cd linux
git checkout v4.4
```

If you return to the `Makefile` now, you should see that the `VERSION` and `PATCHLEVEL` have changed. *Make sure it has, because all the grading will be done using kernel 4.4.0*. Finally, feel free to install in your VM any additional utilities or text editors (e.g., Emacs, Vim) that you find useful.

3 Kernel Compilation

To begin building the kernel, copy the kernel config file from the existing system into your the base of kernel source:

```
cp /boot/config-`uname -r` .config
```

Running `make oldconfig` will then apply the existing system's configuration to this kernel tree. However, in our case there are still many new config options for which there is no existing config file setting. You can press Enter when prompted to accept the default option for each setting. However, to save time you can auto-accept the defaults by running:

```
yes '' | make oldconfig
```

Clean the kernel source directory with

```
make clean
```

We can now compile the kernel. Best practice when compiling the kernel is to parallelize the build, one thread per available core plus one. The below `make` command automatically detects the available CPUs on your VM to do this. Also, to simplify installation, the command will build the linux kernel image and linux header files into `.deb` files.

```
make -j `getconf _NPROCESSORS_ONLN` deb-pkg LOCALVERSION=-NETID
```

IMPORTANT: CHANGE THE ABOVE LINE TO REPLACE "NETID" WITH YOUR NETID. The `LOCALVERSION` field is appended to the name of your kernel.

When this command finally returns, you will have new `deb` files one directory up from the kernel source tree. To install the files, run the following commands:

```
sudo dpkg -i linux-image-4.4.0-*_4.4.0-*_amd64.deb
sudo dpkg -i linux-headers-4.4.0-*_4.4.0-*_amd64.deb
```

4 Booting into Your New Kernel

Now it's time to boot into your newly built kernel. To do so, first you will need to edit your VM's `grub` settings so that the bootloader is visible during the startup sequence. As root, open `/etc/default/grub` with your preferred text editor, then comment out the `GRUB_HIDDEN_TIMEOUT` and `GRUB_HIDDEN_TIMEOUT_QUIET`

lines using a #. Finally, to update your settings run:

```
sudo update-grub
```

The kernel selection screen will now be visible by default for 10 seconds during the boot sequence. You can also set your custom kernel as the default grub entry, but it is recommended that you leave the kernel selection screen open on your VM in case you accidentally brick your custom kernel.

The easiest way to boot into your new kernel will be through opening up a VNC console to the VM. To do so, head to `https://vc.cs.illinois.edu/vsphere-client`, enter in your university NetID and password, and select your VM from the "Navigator" pane. At the top of the central pane that opens, right next to your VM's name, is a button to open up a VM console in a separate window.

Once open, reboot your VM with `sudo reboot` from either the console or from an SSH terminal. In the console, during the boot sequence you will see a Text UI screen titled "GNU GRUB version 2.02 . . . " appear. First, press one of the arrow keys on your keyboard to stop the bootloader from timing out and selecting the default boot option. Then, select "*Advanced options for Ubuntu". Select your custom kernel from the list (*Note: Do *NOT* the one that says "Recovery Mode"*). Verify that everything is working properly by SSHing back into the VM.

5 Submission Instructions

This is an easy one – simply leave the custom kernel running on your VM until you are instructed to do otherwise. The teaching staff will connect to you machine and verify that your custom kernel is running.

6 Running your own local VM

Because we have had problems in the past with the availability of the Engr-IT cloud, it is recommended that you set up a kernel develop environment on your personal machine (e.g., with VirtualBox) by repeating these instructions. Always remember that grading will be based on the Engr-IT cloud, to test your code there before submitting future assignments.