CS423 Fall 2022
MP0: Setting Up a Kernel Development Environment
Due September 13th at 23:59 CDT

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 need a Virtual Machine to complete all the MPs in CS423. You should try to set up your own vritual machine locally, as this allows you to have full control of your VM. In case you cannot set up your own VM, we can help you to request an VM from Engineering IT's VM farm – but do note that if you crash your VM with some bad kernel code, in the worst case you will need to wait for IT to fix it for you, a process which is usually measured in days.

To summarize, you will use one of the following setup:

- A Linux bare-metal machine + QEMU using our QEMU script. This is the most recommended setup, since it allows you to easily gdb your kernel code.

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

- A Mac / Windows machine with a VM client (e.g. VMWare Fusion or VirtualBox). We understand that many of you do not use a Linux machine.
- Engineering IT VMs

P.S.

- In the later steps we assume the use of Ubuntu as your distro.
- If you use a Linux bare-metal machine, you should perform the later steps on your host system, and you do not need the kernel installation step. IF you do not have a Linux bare-metal machine, then you need to perform the later steps inside you VM.

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

```
sudo apt-get install git libncurses-dev libssl-dev libelf-dev dwarves flex
bison
```

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

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

You can check the version of your clone kernel source via the `make kernelversion` command. For CS423, we will use kernel version 5.15.63 (i.e. the latest LTS release at this point).

```
cd linux
git checkout v5.15.63
```

If you execute the `make kernelversion` again, you should see 5.15.63 being printed. *Make sure it has, because all the grading will be done using kernel 5.15.63.*

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, we need to first configure it.

If you are using bare-metal Linux and QEMU, you should also clone the provided qemu scripts and kernel config and the copy the config over:

```
git clone
https://github.com/cs423-uiuc/qemu-script.git ../qemu-script
cp ../qemu-script/.config .config
```

If you are in a VM, copy the kernel config file from the existing system into your the base of kernel source:

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

To apply the kernel config, do:

```
make olddefconfig
```

This will create a config for your kernel based on the old .config file. In case there are any new config without values, it will use the default automatically.

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.

```
make -j`nproc` 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 kernel image built in your kernel directory.

If you are using bare-metal Linux and QEMU, you can now try to boot QEMU from your kernel source root directory with the following command:

```
../qemu-script/cs423-q
```

If everything is correct, you should have a shell inside your VM. You can double check the kernel in the VM the command below and should see your own kernel version (the one with your NetID).

```
uname -a
```

For these of you using a VM client, you need to install your kernel:

```
sudo make headers_install INSTALL_HDR_PATH=/usr
sudo make modules_install -j‘nproc‘
sudo make install
```

4 Booting into Your New Kernel (VM client only)

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 – all you need to submit is the `mp0.txt` from the following command:

```
# in your VM/QEMU
dmesg | grep 'Linux version' > mp0.txt
```

Just keep in mind if you don't do this MP correctly, you will not have able to work on the MPs.