Setup

What to Use

- Lab machines in MCH 202
  - If you filled out the survey, you should have swipe card accessed
    - If you missed it or its not working, go to the Systems Group
  - Email me <dennis@cs.fsu.edu> group layout by February 21
    - No more than 3 people per group
    - Also email me if you are alone and looking for a group
  - We only have 33 machines for 88 students
    - So groups may be merged or split
    - Preference will be given to those who emailed me first
  - I'll reply with the machine number and root password
    - You must implement Project 2 on that machine
    - I recommend adding a new user with admin privileges
      - In case someone else uses the machine later (e.g. project 3, another class)
- For project, use kernel version 4.14.12
  - Download from Canvas

Initial Setup

```
$ sudo apt install libncurses5-dev
$ sudo apt install libelf-dev
$ sudo apt install openssl
$ sudo apt install libssl-dev
```

Downloading the Kernel

Download the linux kernel from Canvas

Extract it in ~/

Rename the directory to test_kernel

```
$ sudo mv “~/test_kernel” “/usr/src/”
$ ln -s “/usr/src/test_kernel” “~/test_kernel”
$ cd “~/test_kernel”
```

Compiling the Kernel

```
$ make menuconfig
```

- Graphical configuration setup
- Stored in .config

```
$ make
```

- Compiles source tree
- Can take hours depending on machine and configuration

```
$ sudo make modules_install
```

- Installs module binaries into modules/

```
$ sudo make install
```

- Installs final binary into /boot

make menuconfig

- Goal is to remove as much as possible without making it unbootable
  - Reduces the resulting binary size and decreases boot time
- Each item has a tristate

[*]	Installed in the kernel directly
[M]	Installed as a kernel module
[ ]	Not installed at all

- Good candidates for removal are device drivers and file systems you won’t use
  - *lspci* to view hardware devices
- Module candidates are things that you may need later but don't warrant loading every time
  - You'll probably have very few of these as you're doing debugging on a static environment
- Include everything else directly in the kernel

- If overwhelmed, just use an old, working configuration
  - I'm not grading your ability to install a stripped down kernel
  - But the more things in the kernel, the longer it will take to compile and install

make oldconfig

- Uses an old configuration to build a new one

```
$ cd “/usr/src/test_kernel”
$ cp “.config” “.config_old”
$ cp “/boot/config-4.13.0-26-generic” “.config”
$ make oldconfig
```

- Accept changes, then you can use `*make menuconfig*` to edit this down
  - Can automate this with `*/bin/yes* | *make oldconfig*`
- This will get you a working setup in case things go wrong

Some Debugging Tools

- Turn on the following in kernel hacking (in make menuconfig):
  - Enable extra timekeeping sanity checking
    - Makes it easier to find execution ordering
  - Lock debugging/*
    - Helps find deadlocks
  - Kobject debugging
    - Includes extra information about the object in the syslog
  - Debug linked list manipulation
    - Adds additional checks when iterating over lists
  - Trigger a bug when data corruption is detected
    - Immediately crashes the kernel when a data structure becomes corrupt
- Turn off the following in kernel hacking:
  - Panic on oops
    - Restarts machine on kernel crash, makes it difficult to see what the crash was

Booting the Kernel

- $ sudo vi /etc/default/grub
  - To change any settings in the boot loader
  - Change "*GRUB_HIDDEN_TIMEOUT=0*" to "*#GRUB_HIDDEN_TIMEOUT=0*"
- $ sudo update-grub
  - Updates the boot loader's binaries
  - Need to do any time you install a kernel or make configuration changes
- $ sudo shutdown -r 0
  - Restart the computer to boot into the new kernel
  - New kernel should be default, but you can always use the "Advanced Options" to find a specific version

Booting Problems

- What would happen if you set all the disk drivers as modules… you wouldn't be able to boot into your kernel
- This is because
  - The boot loader loads the kernel image from /boot
  - The kernel then takes over, but doesn't know how to find /
- Solutions
  - Try each disk driver one by one until you can't boot
    - Time consuming when configuring kernel
  - Include them all
    - Wasteful (time consuming when booting)

When it Doesn't Boot

- Load original kernel
  - You should always have at least one working kernel
- Check that you didn't skip any steps
- Try adding some features back in
  - Use make oldconfig if things get really bad
  - Add things one at a time