Kernel Programming

- **Kernel**: is central part of OS. Computer program that connects software to hardware. Handles:
  - CPU scheduling
  - Memory management
  - Device management
- Kernel code lies in main memory. Develop modules seperately and load as needed
- Linux kernel modules (.ko files)
  - **insmod**: insert module
  - **rmmod**: remove module
- Basic kernel example. Refer to code 1
  - import libraries
  - define which function to
    - initilize(module_init): function that is used as entry point
    - exit(module_exit): function that is used to exit
  - printk is print statement
  - requires Makefile and Kbuild file
- Makefile: organize code compilation
  - Tutorial Makefile explanation. Refer to code 2
- Kernel modules difficult to debug
  - Use a VM
- Kernel API
  - kalloc: malloc for kernel
  - kfree: free for kernel
- **Spinlock**: prevents thread from exectuing code in critical section if it is being executed by another thread. Helps with concurrency
  - #include <linux/spinlock.h>
  - spin_lock_init(&myspinlock)
  - spin_lock/unlock(&myspinlock)

Assignment 1

- max group size of 3 people
- System Call Table
- hijacking a **system call**: service of kernel, log a message and then continue
  - MY_CUSTOM_SYSCALL
  - __NR_exit_group
- have to write 5 functions for this assignment
  - recommendation: start with init_function and exit_function, then move on to the others
- we need to keep track of process IDs and the system call table
- locks are important and look at all the commands involving locks covered in this tutorial
- run tests on VM

```
// Code 1
#include <linux/kernel.h>   // Mark up function eg. __init __exit
```

```
#include <linux/init.h>     // loading kernel modules into kernel
#include <linux/module.h>   // types, macros, functions for kernels

static int mymodule_init(void) {
  return 0;
}

static void mymodule_exit(void) {
  ;
}

module_init(mymodule_init);
module_exit(mymodule_exit);
```

```
# Code 2
# assign variable KDIR
KDIR=/lib/modules/`uname -r`/build

# executed by make kbuild
# -C means change directory to KDIR
# variable M tells where the actual project files are
kbuild:
  make -C $(KDIR) M=`pwd`

# executed by make clean
clean:
  make -C $(KDIR) M=`pwd` clean
```