Driver Basics, part 2

Getting our hands dirty

ECE 373

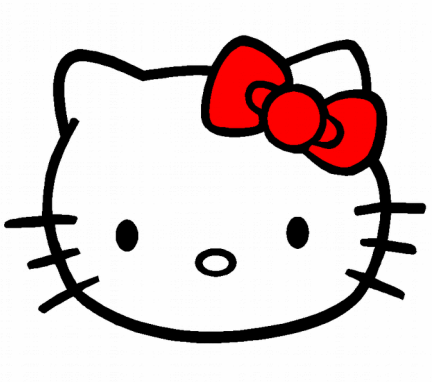
Hello Kernel

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("Dual BSD/GPL");

static int __init hello_init(void)
{
	printk(KERN_INFO "Hello, kernel\n");
	return 0;
}

static void __exit hello_exit(void)
{
	printk(KERN_INFO "Goodbye, kernel\n");
	return 0;
}

module_init(hello_init);
module_exit(hello_exit);
```



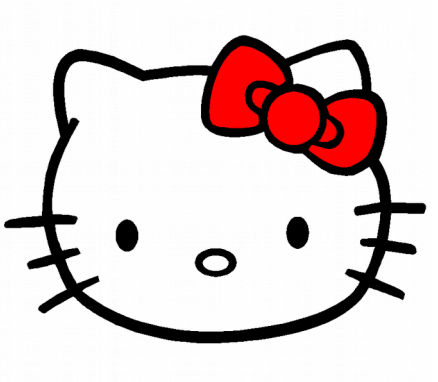
Hello Kernel

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("Dual BSD/GPL");

static int __init hello_init(void)
{
	printk(KERN_INFO "Hello, kernel\n");
	return 0;
}

static void __exit hello_exit(void)
{
	printk(KERN_INFO "Goodbye, kernel\n");
	return 0;
}

module_init(hello_init);
module_exit(hello_exit);
```



Try it out

- Compile: make
- Load: insmod hello_kernel.ko
- Results
  - lsmod
  - dmesg – more detailed, but limited circular buffer
  - tail /var/log/{messages|syslog} – managed syslog text file, not as much detail
- Remove: rmmod hello_kernel

Cross compile

- Build on one machine, load on another
  - Atom box slow, shared env, little disk, no GUI, etc
  - Other embedded systems have similar issues
- Pro
  - Faster compile machine
  - Better development environment
  - Some targets have no native devel env
- Con
  - Copying to target can be tedious
  - Source code not on box for live debug

Cross compile example

- Set up build environment
  - Install build headers
  - Install target compilers?
  - For us, just copy the linux headers

    On target:
    - cd /usr/src
    - tar cvf ~/headers.tar linux-headers-<version>*
    - scp headers.tar me@workmachine:

    On work machine:
    - tar xvf headers.tar
    - Edit KERNEL_DIR to local directory

      KERNEL_DIR ?= ~/linux-headers-<version>-generic
    - make
- Copy result to target and install

  scp hello_kernel.ko target:

  ssh user@target

  sudo insmod hello_kernel.ko

What about the VM?

- What it looks like
- Why, why not
- VMware vs VirtualBox vs virt-manager/qemu
- Which Linux to install