# ECE 373 Assignment #7
# Spring 2016

## Builders of Cata...er...the Kernel

Writing and compiling drivers outside the kernel tree is handy. However, knowing how to integrate it into a kernel tree and build it will lead you one step closer to kernel inclusion, and getting it onto a potential development board.

## Building and transferring the kernel

In order to build your kernel for the Atom machine, you will want to find your build target (unless you have 4+ hours to spend building on the Atom). The ECE Linux lab machines are Ubuntu machines with enough CPU horsepower to build a kernel, so this might be a good place to start if you don't have access to another Linux machine that can build kernels relatively quickly; however, there may be issues with diskspace...

For this part of the assignment, these are the steps you want to perform:

1. Go to kernel.org and download the latest stable Linux kernel tarball (linux 4.6)

2. Unpack the tarball, then copy in the /boot/config-xx file from the Atom box to '.config' and create a default configuration (make olddefconfig).

3. Using one of the configuration edit tools (config, menuconfig, xconfig, gconfig, etc.), find the option to add a Local Version string to the kernel version string, and add your ODIN account name as that string.

4. (**OPTIONAL**) While in the configuration editor, you might choose to disable some options that you know aren't used in the Atom box as a way to make the compile go faster and the resulting files be smaller. For example, you probably don't need Virtualization or Linux Guest support, nor do you need support for so many filesystem or networking types or drivers.

5. Now build the kernel into a Debian package (what Ubuntu uses), by using the 'make bindeb-pkg' build target (remember the -j option to make to speed up the build!). Remember that 'make bindeb-pkg' can only be done on an Ubuntu machine, so plan accordingly.

6. Transfer the built kernel packages to the Atom machine (probably via USB stick) and install them using the dpkg command (man dpkg...)

NOTE: Depending on the grub2 configuration, you may need to select your kernel image on boot. This will require either a monitor and keyboard to be connected to the Atom machine, or you need to be connected via serial console when the machine powers on so you can select your kernel from the boot menu.

## Add your driver!

This final part of the assignment is to add your driver you've been writing this term to the kernel tree. This will involve adding a new Kconfig and Makefile for your code.

1. Create a new directory for your driver in drivers/misc called ece

2. Copy your kernel driver from any of the kernel driver assignments into this directory (minus the Makefile)

3. Now create a Kconfig file for your driver with the following attributes:

    a. Have a decent description of the driver for the selection menu

    b. Make it a tri-state

    c. Make it depend on PCI

    d. Include a somewhat useful help text for the driver, what it does, etc.

4. Now add a simple Makefile, and edit the drivers/misc/Kconfig file to incorporate your new driver into the build environment (refer to another driver for reference, the drivers/misc/mei driver is a good place to look)

5. Now re-run your config option of choice (config, menuconfig, xconfig), and find and select your driver as a module.

6. Re-build the kernel from the first part of the assignment (make deb-pkg), transfer it to the Atom, and boot your new kernel with your driver.

## How to finish

Turn in these materials before or at the start of class on **Tuessday**, **7-June-2016:**

1. The drivers/misc/Kconfig, drivers/misc/ece/Kconfig, and drivers/misc/ece/Makefile, and drivers/misc/Makefile

2. A typescript showing you executing uname -r after you booted your new kernel (showing your modified kernel version string)

3. A typescript showing the ls listing of /lib/modules/`uname -r`/kernel/drivers/misc/ece showing your kernel object built and in the kernel

4. A typescript showing the lsmod output showing your kernel module loaded on this new kernel

5. A note from someone (anyone, but Vijay is a good choice) saying they actually saw your booted kernel and your driver loaded from the in-kernel tree