Building a custom kernel

For the upcoming homework #3, you will be asked to implement some new functionality in the Linux kernel. For that, you need to know how to build a custom kernel from source.

This tutorial shows you how to build a custom kernel on Fedora Linux. If you have not built a Linux kernel from source code before, you can try to follow the tutorial and hopefully you will know how to do it in the end.

Note that there are many webpages on the Internet teaching people how to build Linux kernel from source code. You are encouraged to look up those pages, too, in case the information in this tutorial is outdated or may you prefer a different Linux distribution (e.g. Ubuntu), in which case the steps for building custom kernel are slightly different.

A. Make sure the system has all the necessary packages installed.

```
su -c 'yum install rpmdevtools yum-utils'
su -c 'yum install qt3-devel libXi-devel'
```

B. Prepare a RPM package building environment in your home directory.

```
rpmdev-setuptree
```

C. Get the source.

```
yumdownloader --source kernel
```

At your home directory, you should have the kernel source rpm and the rpmbuild directory as follow:

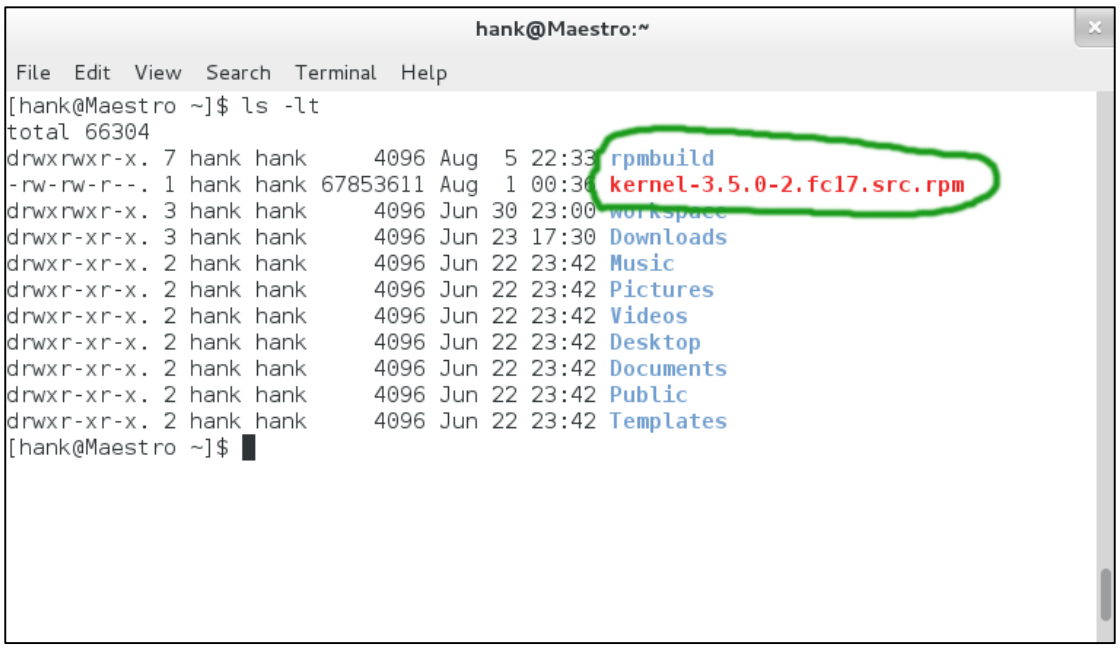

Figure 1. kernel rpm and rpmbuild directory

D. Install build dependencies for the kernel source

```
su -c 'yum-builddep kernel-<version>.src.rpm'
```

Note that you need to replace <version> with the version of the kernel source you downloaded. For instance, the version of the kernel in Figure 1 is "3.5.0-2.fc17".

E. Install the kernel source

```
rpm -Uvh kernel-<version>.src.rpm
```

This command writes the RPM contents into `${HOME}/rpmbuild/SOURCES` and `${HOME}/rpmbuild/SPECS`, where `${HOME}` is your home directory.

F. Prepare the kernel source tree

```
cd ~/rpmbuild/SPECS
rpmbuild -bp --target=$(uname -m) kernel.spec
```

The kernel source tree is now located in the `~/rpmbuild/BUILD/kernel-<version>/linux-<version>.<arch>` direct

ory. For instance, the kernel source tree directory should look like:

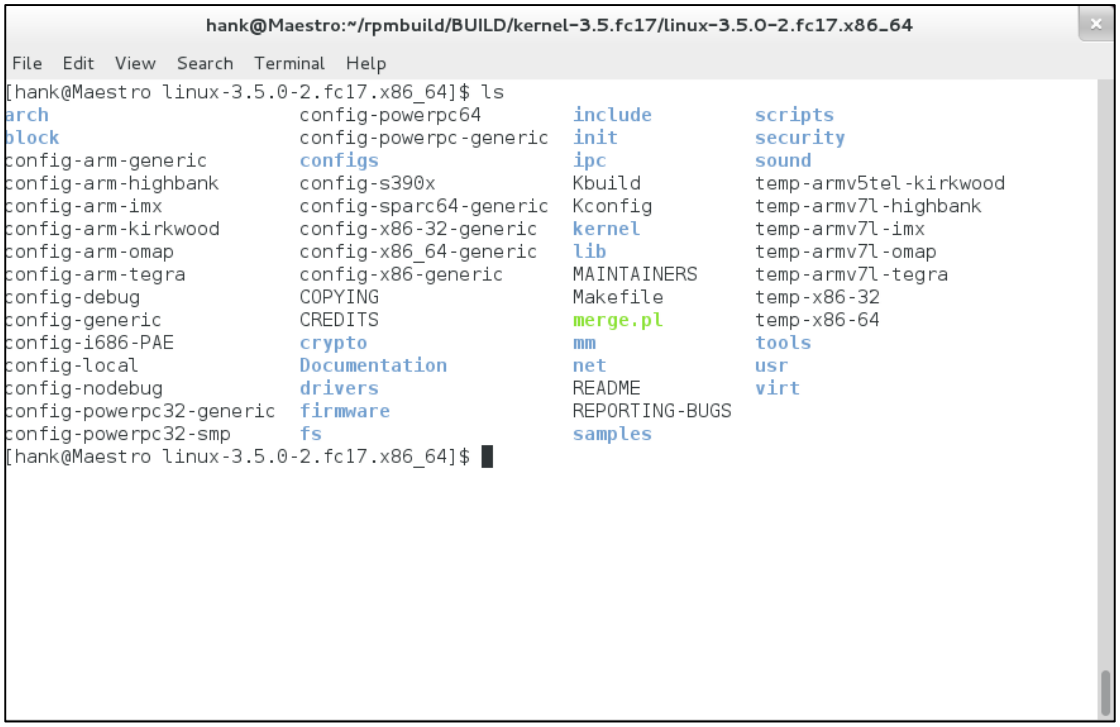

Figure 2. Kernel Source Tree

G. Configure Kernel Options

```
cd ~/rpmbuild/BUILD/kernel-$ver.$fedver/linux-$ver.$arch/
make xconfig
```

H. Give your kernel a unique name by changing EXTRAVERSION in Makefile

hank@Maestro:~/rpmbuild/BUILD/kernel-3.5.fc17/linux-3.5.0-2.fc17.x86_64

File Edit View Search Terminal Help

```
VERSION = 3
PATCHLEVEL = 5
SUBLEVEL = 0
EXTRAVERSION = _OS_COURSE_1234567
NAME = Saber-toothed Squirrel

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in ./README
# Comments in this file are targeted only to the developer, do not
# expect to learn how to build the kernel reading this file.

# Do not:
# o  use make's built-in rules and variables
#    (this increases performance and avoids hard-to-debug behaviour);
# o  print "Entering directory ...";
MAKEFLAGS += -rR --no-print-directory

# Avoid funny character set dependencies
unexport LC_ALL
LC_COLLATE=C
LC_NUMERIC=C
export LC_COLLATE LC_NUMERIC

# We are using a recursive build, so we need to do a little thinking
# to get the ordering right.
```

Figure 3. Customizing kernel name

I. Build Kernel Image

```
make bzImage
```

If the build is successful, you should see the following screen

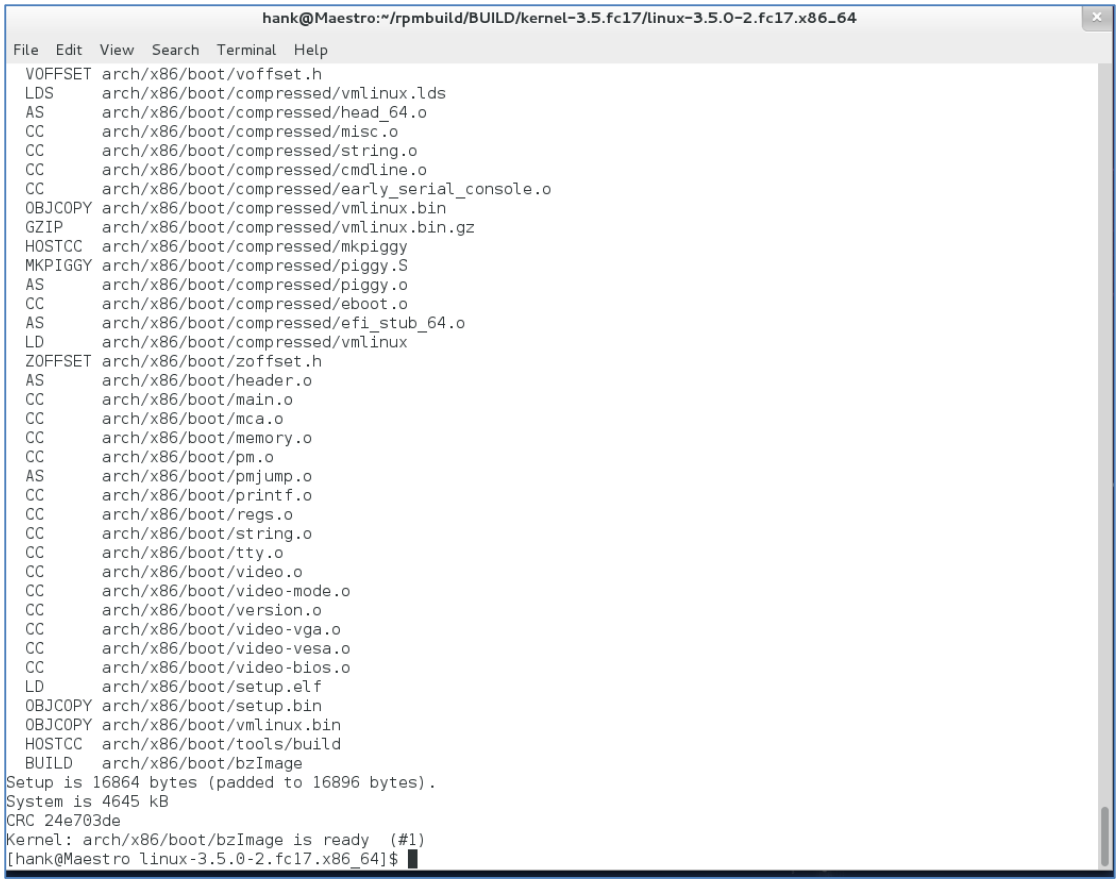

Figure 4. Build kernel succesfully

J. Build modules

```
make modules
```

If the build is successful, you will see the following screen

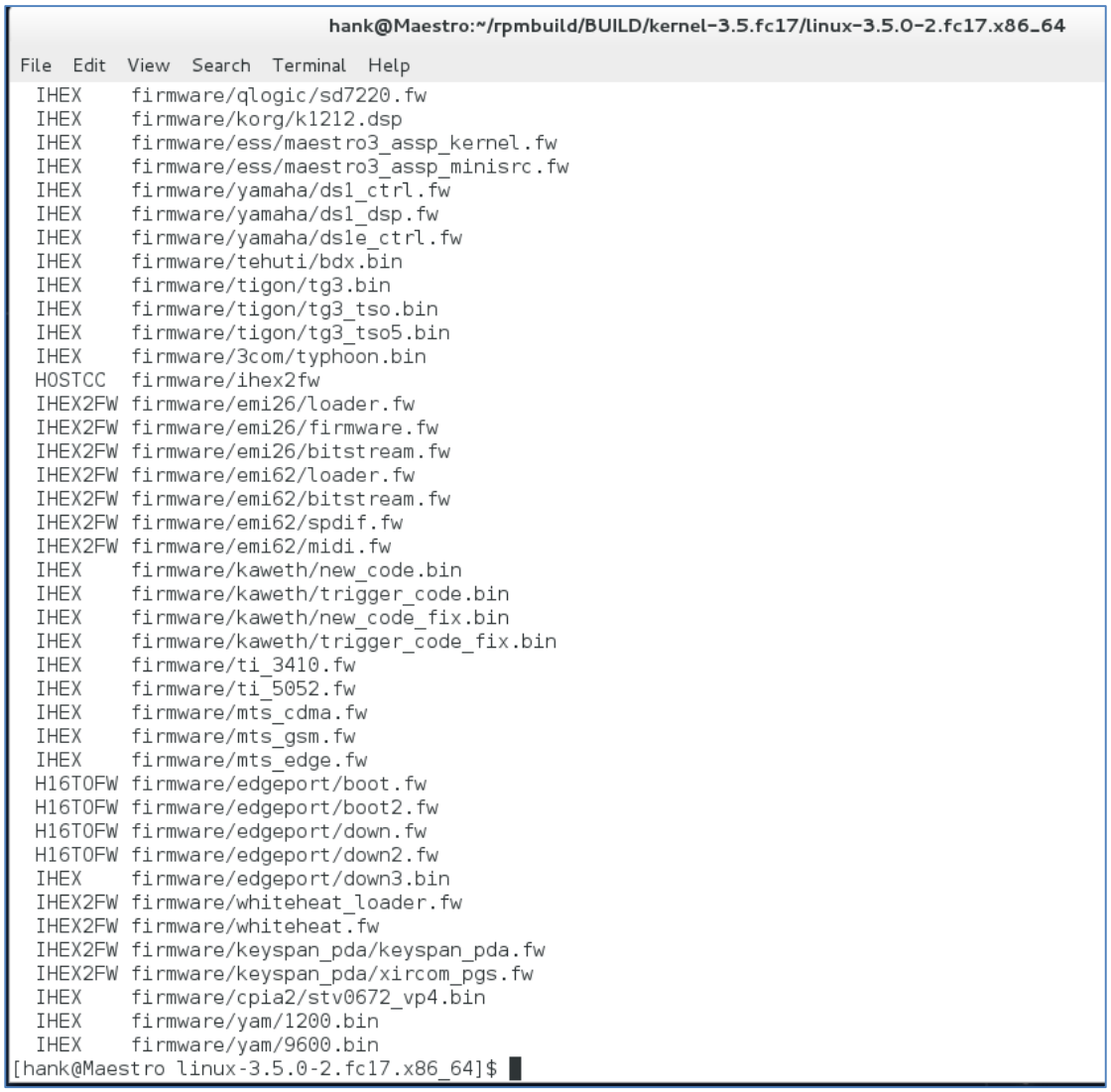

Figure 5. Build modules successfully

K. Install modules

```
su –c ‘make modules_install’
```

Modules will be installed under /lib/modules. For instance, following the example above, the modules will be installed under /lib/modules/3.5.0_OS_COURSE_1234567 (Figure 6).

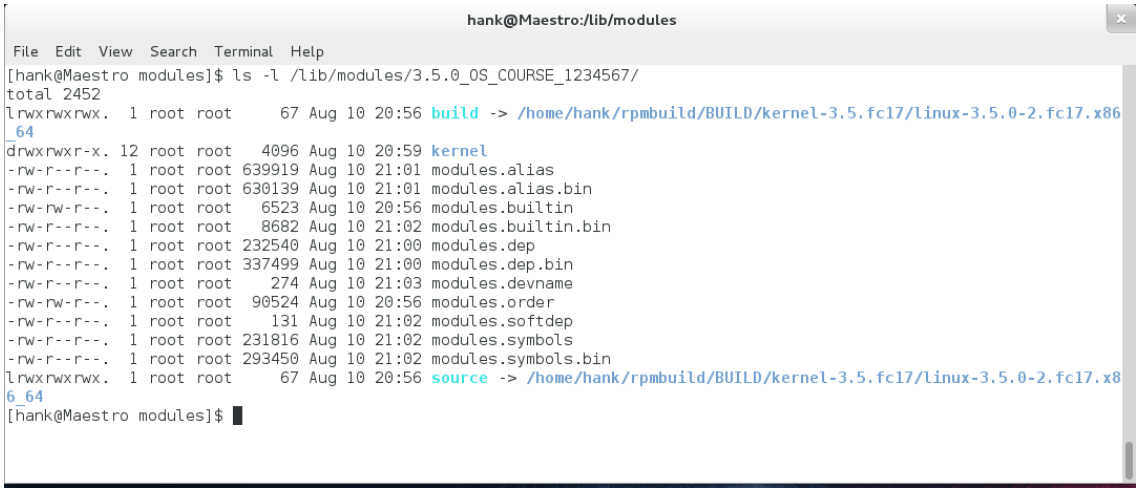

Figure 6. Kernel modules installation location

L. Install kernel image

su –c 'make install'

The kernel image will be installed under /boot as shown in Figure 7.

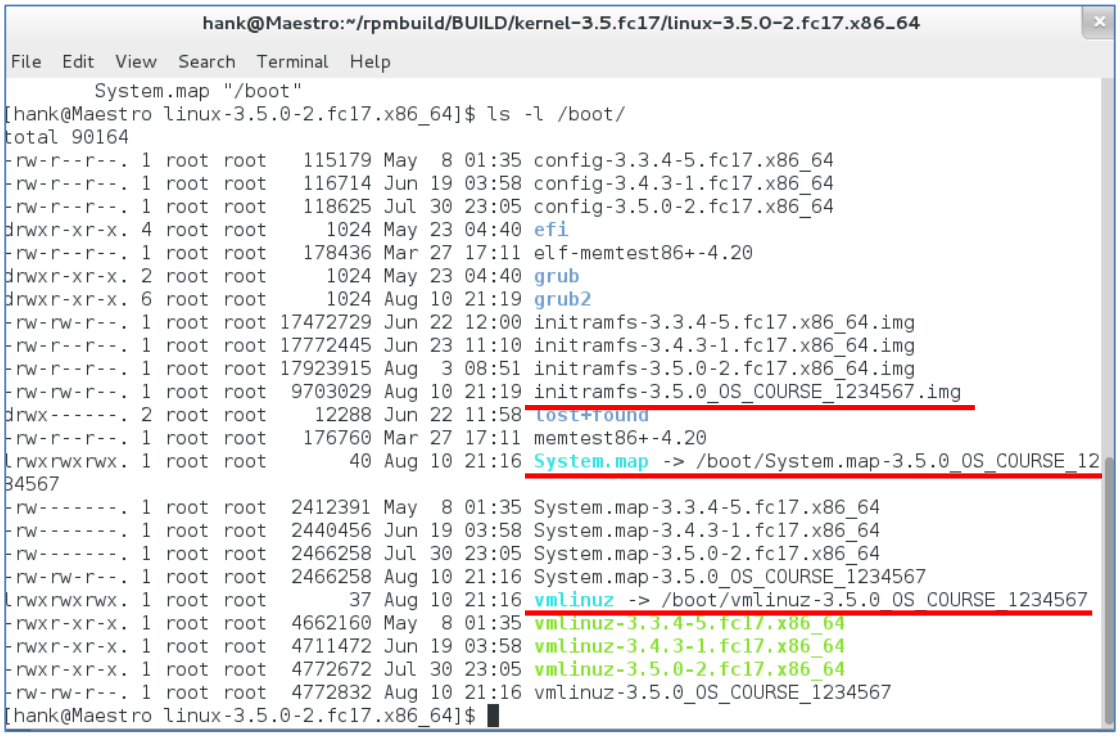

Figure 7. kernel image installation location

M. Boot with the new kernel