CSE 237A Project: Compile Guide

This guide will provide you with step-by-step instructions on how to compile the kernel and entire Android system, as well as how to merge your custom kernel into the image for flashing the phone.
If any mistakes were made during the setup of the project or if any tools are missing, the compilation process will give an error at some point and fail. As long as everything was setup correctly, the following commands will compile the Android system and generate an image file that is ready for flashing onto a phone.

Compiling: Kernel

The first step is to simply compile the kernel. The first step is to make one change to .config found in ~/android/htc-kernel-msm7227/.
Open the file `~/android/htc-kernel-msm7227/.config` in a text editor and then change the line

- `CONFIG_USB_ANDROID_PROJECTOR=y`

to

- `# CONFIG_USB_ANDROID_PROJECTOR is not set`

If you forget to do this, you will get an error that looks like:

> drivers/usb/gadget/f_projector.c: In function 'projector_bind_config':
> drivers/usb/gadget/f_projector.c:759: error: 'struct usb_function' has no member named 'hidden'

We will now make a change to the kernel necessary for your kernel module to work correctly during the project. Download the file `sched.c` from the course website and place it in `~/android/htc-kernel-msm7227/kernel/`, replacing the existing `sched.c` located there.
To compile the kernel:

- `cd ~/android/htc-kernel-msm7227/`
- `make`

When compilation is complete, the file zImage will be generated and located at `~/android/htc-kernel-msm7227/arch/arm/boot/zImage`.

Compiling: Android

The following steps will configure the build process for your particular device (HTC Liberty). If they are not completed correctly, the compilation process may try to build the system for the wrong hardware and fail.

Certain proprietary drivers are required for the phone to function correctly. First:

- `mkdir -p ~/android/system/vendor/htc/liberty/proprietary`

Then download drivers.tar.gz from the course website and extract the files to `~/android/system/vendor/htc/liberty/proprietary`. You should **NOT** be creating a directory called `drivers` inside the `proprietary` directory. All of the files should be placed directly into the `proprietary` directory. Then:

- `cd ~/android/system/device/htc/liberty/`
- `./extract-files.sh`
    - This command will give a bunch of identical errors about files not being found. You may ignore this. The other commands in the script that DO work are providing no output. This will work fine as long as you remembered to extract the drivers correctly above.

- `cd ~/android/system/`
- `cp ../htc-kernel-msm7227/.config .config`
    - This grabs a copy of the .config file you made changes to earlier and places it in the current directory
- `cp ./vendor/cyanogen/products/cyanogen_liberty.mk ./buildspec.mk`
- `. build/envsetup.sh`
- `lunch cyanogen_liberty-eng`

At this point, your build should be configured and ready to go! To compile:

- `make -j`grep 'processor' /proc/cpuinfo | wc -l` CYANOGEN_WITH_GOOGLE=true bacon`

This will start the compilation process, which can take about an hour the first time. If everything compiles correctly, you should see a message near the end that the image update-cm-X.X.X-Liberty-signed.zip was placed at ~/android/system/out/target/product/liberty/.

Compiling: Custom Boot

The Android image that you just installed uses a precompiled kernel image. You will now prepare your own custom kernel for flashing over the existing kernel. Run the following commands on your host machine:

- `mkdir ~/android/bootimg`
- `cd ~/android/bootimg`
- `cp ~/android/system/out/target/product/liberty/boot.img .`
- `cp ~/android/htc-kernel-msm7227/arch/arm/boot/zImage .`
- `~/android/system/out/host/linux-x86/bin/unpackbootimg -i boot.img`
- `~/android/system/out/host/linux-x86/bin/mkbootimg --cmdline 'no_console_suspend=1 console=null' --kernel zImage --ramdisk boot.img-ramdisk.gz -o newboot.img`
- `dd if=boot.img of=newboot.img bs=48 count=1 conv=notrunc`

This will create newboot.img, which will be flashed over the existing ‘boot’ partition of your device, thus installing your custom kernel.

Installing: Android Image

To install the compiled image, we will need to push the image to the phone and start the phone in ‘recovery’ mode. While booting up the phone, hold the ‘down volume’ button on the side of the phone. This will boot the phone into fastboot mode. Press the power button to

select ‘BOOTLOADER.’ After a brief delay, press down and up on the volume control tPress the power button to select ‘BOOTLOADER.’ After a brief delay, press down and upo move the cursor up and down, and press the power button to select ‘RECOVERY.’ The phone will then boot into recovery mode. While in recovery mode, on your host machine do the following:

- `cd ~/android/system/out/target/product/liberty/`
- `adb push update-cm-X.X.X-Liberty-signed.zip /sdcard/myImage.zip`
    - where XXXXX is the Cyanogen version number

The image will now be on the SD card of the phone. On the phone, now perform the following steps:

- Select ‘wipe data/factory reset’ and then ‘Yes’
- Select ‘wipe cache partition’ and then ‘Yes’
- Select ‘install zip from sdcard’
- Then select ‘choose zip from sdcard’ and select ‘myImage.zip’ (the image you pushed earlier) and then select ‘Yes’
- It will take a moment to extract your image and flash it to the device

Next you will install Google applications, such as Google Maps, that may be interesting to test and mess around with during your project. To do so, download from the course website gapps-mdpi-20101020-signed.zip to any location on your computer, and from the directory containing the file run:

- adb push gapps-mdpi-20101020-signed.zip /sdcard/gapps.zip

Then on your device:

- Select ‘choose zip from sdcard’ and select ‘gapps.zip’ and then select ‘Yes’

After all of the above steps have been completed, select ‘reboot device now’ on the phone. If everything has been completed and is working correctly, the phone should automatically begin loading and booting into Android!

Installing: Custom Kernel

These steps will flash over the installed boot partition with newboot.img that contains your custom kernel. Reboot the phone into recovery mode and do the following.
Push the image to the device:

- `adb push newboot.img /sdcard/newboot.img`

And finally, install the image on the device:

- `adb shell` (this will open a shell to the phone)
- `cat /dev/zero > /dev/mtd/mtd2`
    - This will eventually return an error that No space is left on the device. You can ignore this error.
- `flash_image boot /sdcard/newboot.img`

After this is complete and you reboot your device, Android should start up and be running your custom kernel!

If your device reboots over and over again or if it locks up, there is something wrong with your newboot.img. You may need to remove the battery from the device, and hold down volume key while putting the battery back in to get it to start into recovery mode again.