

WES 237A Project Part 1 Guide

A. Environment Setup Guide

Goals

The purpose of this document is to prepare your development machine for the project by:

1. Installing any missing, required tools
2. Setting required environment variables and paths

Reporting Errors

If any step stop prematurely and shows an error message, do not proceed forward. Log your error message and *READ THE ERROR MESSAGE*. Some common errors can be figured out with a quick web search. If you need assistance, be sure to include your error message and any relevant details when posting on the class discussion board or when emailing the TA.

Here is an example how to log the error message:

```
[your_command] 1>log 2>error
```

After you execute this line, you should have two files 'log' and 'error', 'log' is the output of your command, while 'error' is the error message.

Disclaimer

If you are new to working in a Linux environment, please be sure to take the time to learn some basics on your own. This class assumes that all students are comfortable with C development on Linux. Some basics can be learned from the following guide:

<http://code.google.com/edu/tools101/linux/basics.html>

Setting up your development environment

Setup

This guide was tested on a machine running Ubuntu 64-bit 10.04. It should work fine for any recent version of Ubuntu as well, but with later versions, your mileage may vary. If you have only Windows machine, we strongly recommend you to install a dual boot Ubuntu by using Ubuntu 10.04 Windows Installer (see instructions [here](#)), or please refer to the Ubuntu Virtualization Guide posted on the course web site for steps on how to get Ubuntu running on your machine in a virtualized environment.

Installing tools:

In a Linux Terminal, enter the following commands (you can copy/paste single lines, but *DO NOT* copy/paste multiple lines, you will trigger execution of multi-line commands:

```
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
```

```
sudo apt-get update
```

```
sudo apt-get install git-core gnupg vim flex bison gperf build-essential \  
zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \  
x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev \  
libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \  
libxml2-utils
```

```
sudo add-apt-repository ppa:ferramroberto/java
sudo apt-get update
sudo apt-get install sun-java6-jdk

mkdir ~/bin
```

Downloading the adb:

The Android Debug Bridge is included with the Android SDK, but to simplify the installation process we have posted the adb binary for linux on the course website. Please download it and place it in ~/bin.

Setting up environment variables:

Edit or create the file ~/.bash_aliases, and add the following line:

```
alias sudo='sudo env PATH=$PATH'
```

Edit the file ~/.profile, and add the following line:

```
PATH=".:$HOME/UCSD-CSE237A-Android/out/host/linux-x86/bin:$HOME/bin:$PATH"
```

In a terminal, enter the following:

```
source ~/.bash_aliases
source ~/.profile
```

To verify that your path is set up correctly, try the following command in your terminal:

```
echo $PATH
```

You should see output that looks something like (but necessarily exactly like):

```
./home/<USERNAME>/UCSD-CSE237A-Android/out/host/linux-x86/bin:/home/<USERNAME>/
bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Where <USERNAME> is your username.

At this point, you are done with Environment Setup.

B. Project Setup Guide

Goals

The purpose of this document is to:

1. Download all required source code
2. Compile android kernel and system
3. *Connect to the MSM8660 platform and flash the device with your compiled system image*

NOTE FOR WES 237A students: Do NOT continue to sections in gray italics!

Install repository tools and download the source

To install the repository tools, do the following in your terminal:

```
sudo apt-get -y install git-core
```

```
cd ~
```

```
mkdir mybin
```

```
curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/mybin/repo

cd mybin

chmod a+x repo

export PATH=~/mybin:$PATH
```

Download the Source

To download the source (about 1GB, so plan accordingly), open a terminal and execute the following:

```
cd ~

mkdir UCSD-CSE237A-Android

cd ~/UCSD-CSE237A-Android

repo init -u git://codeaurora.org/platform/manifest.git -b ics_chocolate -m
M8260AAABQNLZA30135.xml --repo-url=git://codeaurora.org/tools/repo.git

repo sync -j4 (or j8, depending on the number of cores on your Linux machine)
```

NOTE: If this download is interrupted, you may resume it by running the command again.

Compile the Source

To configure and compile the source code, do the following in your terminal:

```
cd ~/UCSD-CSE237A-Android

source build/envsetup.sh

choosetool 1 msm8660_surf eng

make -j4 ( or j8 depending on the number of cores on your Linux machine)
```

NOTE: This command takes multiple hours to finish, so make sure your host machine is ready to do this before executing

NOTE: It is important that you do not get any errors during this step. Please make sure the final output shows a successful compilation

Cross-compile the test C file

To verify the tools were built correctly, do the following in your terminal:

```
cd ~

mkdir cross_compile

cd cross_compile
```

Download the `test.c` and `Makefile` files from the class website into this directory. In the terminal:

```
chmod a+x test.c

chmod a+x Makefile

make
```

There should now be a `test.o` file in your `cross_compile` directory. Verify that the cross-compiling has worked with the following command:

```
file test.o
```

The output should be something along the lines of "ELF 32-bit LSB relocatable, ARM, version 1 (SYSV), not stripped". If you instead see a reference to Linux or x86, please recheck that the compilation has worked.

Submission

When complete, submit your `test.o` file on TED.

Device Information

Before flashing the device, please be sure to plug it into a power source and let it charge for a few hours. Do not try to charge the device by plugging it into your computer's USB port. USB charging seems to not work consistently for these devices.

If during your project you notice that the device is no longer visible when you run the `adb` and `fastboot` commands (mentioned below) but it was visible before, do a hard reset on the device.

When you check out the device from the TA, he will talk you through how to reset the device if it gets into a stuck or locked up state.

If you are having trouble with the device, please post on Piazza and come to office hours. Try test your device AS SOON AS POSSIBLE. If something is wrong, it must be resolved early on.

Flashing the Kernel

1. Open a terminal window and execute the following:

```
mkdir ~/.android
```

```
echo 0x0956 >> ~/.android/adb_usb.ini
```

```
echo 0x18d1 >> ~/.android/adb_usb.ini
```

```
echo 0x05c6 >> ~/.android/adb_usb.ini
```

2. Boot your phone

3. On your phone, go to Settings -> Applications -> Development -> Enable USB debugging

4. On your host machine, start `adb` with root privilege

```
sudo killall adb
```

```
sudo adb devices
```

5. You should see onscreen:

```
List of devices attached
```

```
[your_device_ID] device
```

NOTE: If you cannot see any device, stop and check previous procedures (1~3)

6. Reboot your phone to bootloader mode

```
sudo adb reboot bootloader
```

```
sudo fastboot devices
```

7. You should see onscreen:

```
[your_device_ID] fastboot
```

6. Flash your own kernel onto the device:

```
cd out/target/product/msm8660_surf  
sudo fastboot flash boot boot.img
```

8. Reboot your device and let it start up Android!

```
sudo fastboot reboot
```

Once phone is rebooted, you should see the android home screen (image on left). When you go to Settings -> About phone you should be able to see the new kernel version.