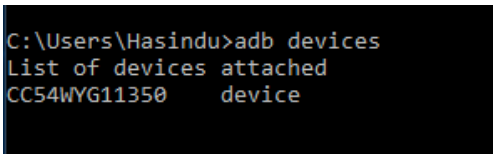
CO224 : Computer Architecture

Testing on a real arm processor – your Android mobile phone

Step 1: Installing the ADB bridge to support your phone

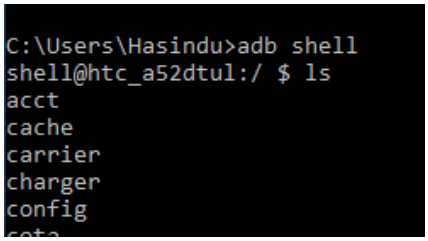
On Windows

- Download the latest version of ADB bridge from the following link
  http://forum.xda-developers.com/showthread.php?t=2317790
- Enable the developer's mode on your device and enable USB debugging (Search in google if you don't know how to do this)
- Connect the mobile phone on to the PC and authorize the connection request on the mobile phone, if it asks.
- Type *adb devices* on a command prompt and you will see something like this.



  If *adb* command is not found you might have to set the path environment variable.

- Type *adb shell* and now you get the shell prompt in your mobile phone, where you can execute most Linux commands you know, on your Android.



On Linux

On Linux you can install ADB bridge by the following commands. After that the steps are same as for Windows.

sudo add-apt-repository ppa:phablet-team/tools && sudo apt-get update
sudo apt-get install android-tools-adb android-tools-fastboot

This is a bit old version and hence if your phone is too new, this would fail to connect to your phone. So best option is to install the Android SDK. You will need to select android platform tools when installing and set the path appropriately. Find help on the following link.

http://askubuntu.com/questions/318246/complete-installation-guide-for-android-sdk-adt-bundle-on-ubuntu

If you don't have an Android phone we will connect an Android device on the tesla server in few days. ssh to tesla and directly you can issue the command *adb shell*.

Step 2: Installing a cross compiler

Unless your phone is rooted it is not possible to install a compiler on to your devices. Hence we will have to use a cross compiler that targets Android. On Ubuntu we can install the cross compiler by the following command.

sudo apt-get install gcc-arm-linux-androideabi

We have already setup this on the tesla server for your convenience.

Step 3 : Compiling and running

- Put the source code to the Linux machine which you have *gcc-arm-linux-androideabi* installed and issue the following command to compile.

  arm-linux-androideabi-gcc -Wall source.s -o binary -fPIE –pie

  Try without *-fPIE –pie* first, as in some phones such as the one we are going to connect to the tesla it works without that. But some phones require Position-independent executables.

- Copy the compiled binary file into the phone. You can use the *adb tool* as follows.

  For example, I would copy *to /data/local/tmp* as that location seem to give proper permissions for the *adb shell*. Issuing the command *adb push binary /data/local/tmp* copies the file binary on the current folder on computer to */data/local/tmp* folder in the Android. Ceratin phones such as the one we are going to connect to the tesla, doesn't have any permission issues in anywhere and hence you can even copy to some folder in the *sdcard*!

```
C:\Users\Hasindu\Desktop>adb push binary /data/local/tmp
601 KB/s (3080 bytes in 0.005s)
```

- To run the binary file use *adb shell* as follows.

```
C:\Users\Hasindu\Desktop>adb shell
shell@htc_a52dtul:/ $ cd /data/local/tmp
shell@htc_a52dtul:/data/local/tmp $ ls
binary
shell@htc_a52dtul:/data/local/tmp $ chmod 777 binary
shell@htc_a52dtul:/data/local/tmp $ ./binary
WARNING: linker: ./binary has text relocations. This is wasting memory and prevents security hardening. Please fix.
The Answer is 5 (Expect 5 if correct)
38|shell@htc_a52dtul:/data/local/tmp $
```

I have tested these on an HTC Desire 826 and the phone we are going to connect to the tesla (An ZTE phone). Can't guarantee these steps might work as is on your phone and hence you might have to bit play around as well to get it working ;)