**Cross Compiling C/C++ for Android :**

Let Say you want to run your c / c++ program in your android mobile phone then follow below given steps :

What you required :

c/c++ file , At work, when working on Android, we typically checkout the entire Android source code (which is huge), use lunch to configure a ton of environmental variables, then use Makefiles with lots of includes and special vars. We don’t want to spend the time and disk space checking out the Android source code just to have a working cross compiler. Luckily, the Android tools team has an excellent utility to grab a prebuilt cross compiler.

**First thing you need Android NDK :**

if you dont have then grab from the below steps run on your machine

**curl -O \** <http://dl.google.com/android/repository/android-ndk-r12b-linux-x86_64.zip>

**unzip android-ndk-r13b-linux-x86\_64.zip** 

It would be helpful to install adb and fastboot, too. This might be different for your distro’s package manager. Better yet may be to just build from source.

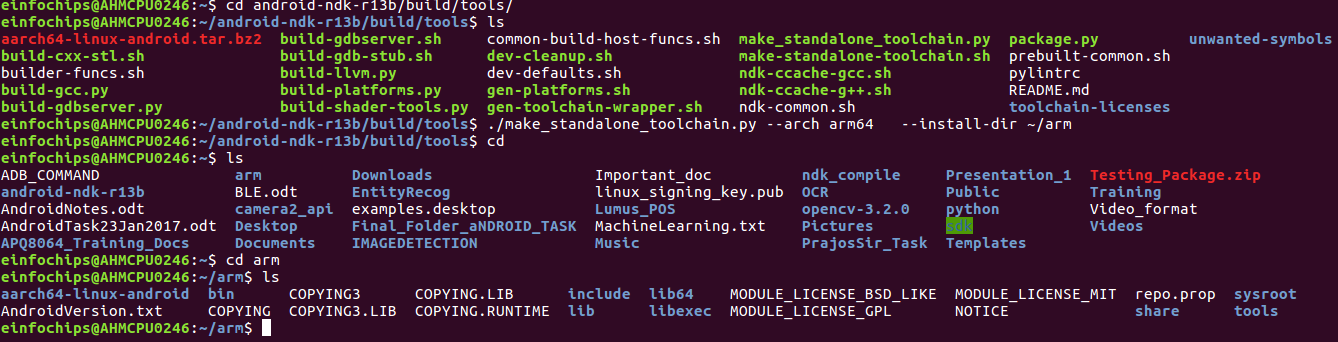
**Sudo apt-get install android-tools-adb android-tools-fastboot**

Once you install all required tools then we have to have arm file according to the android device So According to your android devices we have to Buid toolchain :

**In Your Android-ndk-r13b/build/tools you will find a standalone toolchain we have to  run that script .**

**./make\_standalone\_toolchain.py --arch arm64   --install-dir ~/arm**

After running the script you will see arm folder in your specific path .If you are getting then we are reafy to cross comiple a file in your sepcific device. As Shown in diagram you can follw the steps :



This will create a nice standalone bundle in arm . It will contain our cross compiler, linker, headers, libs, andsysroot(crt.o and friends ). Most Android devices are ARMv7-A, so you’d use –arch arm

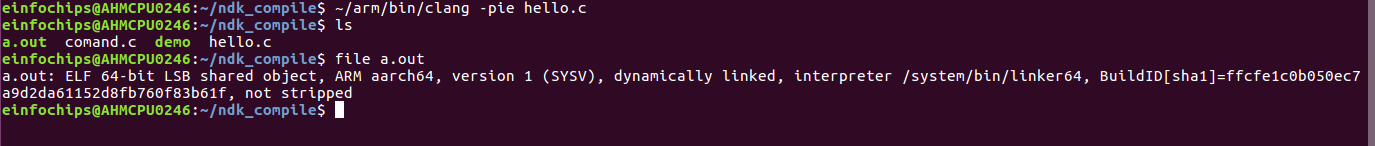
If your device dosent support these arm architecture then we have to

**check https://developer.android.com/ndk/guides/standalone\_toolchain.html#itc**

 By these site we will have specfic device arm architecture You might also want to change your install-dir and possible add it to your $PATH, or set $CC and $CXX.

Now , You are ready to cross comiple your c / c++ code .

Create a c/c++ file then follow the steps :



Explanation :

~/arm/bin/clang -pie hello.c

 Since Android Lollipop Android has required that executables be linked as position independent (-pie) to help provide ASLR.

 Android 4.0 Ice Cream Sandwich and above provides address space layout randomization (ASLR) to help protect system and third party applications from exploits due to memory-management issues

<install-dir>/bin/ also has shell scripts with more full featured names like aarch64-linux-android-clang if you prefer to have clearer named executables in your $PATH.

We get the executable file push the file to the android device Connect your phone, enable remote debugging, and accept the prompt for remote debugging:

Text

Description automatically generated

Next Step is to Cross Compile with Android Source So Stay tuned ..

Second Method :