Android NDK Cross-Compile Setup (libpng and freetype)

21 Tuesday Aug 2012

POSTED BY MORTORAY IN PROGRAMMING, USE CASE

≈ 6 COMMENTS

Tags
android, c++, native libraries, ndk

Building native libraries for Android can be a bit confusing. My goal, which is likely shared by many, was to get libpng and freetyp2 working for my native OpenGL application. It took me a while to find enough references and examples before I was able to get it working. Now that I have a solution I thought I'd share the process.

Build Machine

I'm using Kubuntu 12.04 64-bit as my build machine. I do however suspect any relatively new Linux install will work just fine. You don't actually need much from the build platform as most of the tools are available in the Google NDK, which I'll assume you have installed.

Setup Native Toolchain

The first thing you need to do is create the toolchain needed to build the native binaries. Note that this "toolchain" is simply the setup for GCC to work as a cross-compiler. You can refer to the "docs/STANDALONE-TOOLCHAIN.html" document from the Google NDK, it has good information if you want more details.

We need to create a directory which will host this toolchain, and the resulting libraries. So set these environment variables:

```
PLATFORM_PREFIX=/opt/android-ext/
NDK_PATH=/opt/android-ndk-r8b/
NDK_PLATFORM=android-9
```

The PLATFORM_PREFIX is simply where I want to put the toolchain. Everything in this directory will be for one particular platform. In this case we'll be targetting the default ARM chipset (if you wish to support another one you'll need to duplicate this process with a different PLATFORM_PREFIX).

NDK_PATH simply says where you installed your Google NDK. NKD_PLATFORM says which Android API version you are targetting. You'll need to ensure you have the platform support files for this installed (you should probably already have these if you're doing any NDK work). I choose android-9, as do most examples I've seen, since it is still widely deployed, even on devices released this year.

Now we'll simply use the NDK script to setup what we need there.

6

```
mkdir $PLATFORM_PREFIX

$NDK PATH/build/tools/make-standalone-toolchain.sh  --platform=$NDK
```

Whenever we wish to use this toolchain we add it to the path as below. It is very important that this path is otherwise quite clean: it must not contain other items from the NDK. If you're like me you probably have some of the NDK tools in your path for convenience. During cross-compiling you don't want this, it will break the building. You also want to make sure your current environment is free of anything like CFLAGS, CPPFLAGS, LD_LIBRARY_PATH and other things which might alter how the compiler works.

```
PATH=$PLATFORM PREFIX/bin:$PATH
```

Build libpng

Loading images is probably one of the fundamental things you'll need in your application. It's unfortunate that Google did not expose the native PNG library for use in the NDK. You will thus need your own version. Now that we have the cross-compiler setup this part is actually quite easy — you simply use the package provided configure script.

```
cd /tmp
tar xzf /opt/mortoray.com/source-packages/libpng-1.5.12.tar.gz
cd libpng-1.5.12/
./configure --host=arm-linux-androideabi --prefix=$PLATFORM_PREFIX
make
make install
```

I'm showing all the commands just for clarity. You can grab the libpng file from the source, there is no need to download a special version. Note the use of PLATFORM_PREFIX here indicating where your toolchain is. The "–host" parameter to configure indicates cross-compling to that architecture. (Automake gives a warning about –host, and I'm not entirely clear as to why?)

If this worked you will now find a "libpng.a" file in your "$PLATFORM_PREFIX/lib" directory. You can use this as prebuilt external library in your "Android.mk" file now. I've provided a rough example from my code below.

```
PLATFORM_PREFIX := /opt/android-ext/

LOCAL_PATH := $(PLATFORM_PREFIX)/lib
include $(CLEAR_VARS)
LOCAL_MODULE := libpng
LOCAL_SRC_FILES := libpng.a
include $(PREBUILT_STATIC_LIBRARY)

# The in your project add libpng
LOCAL STATIC LIBRARIES := android native app glue libpng
```

Build libfreetype

I was able to use the exact same process for libfreetype. In examples I've seen all sorts of variations and flags which are needed to get it to work, yet I've had no problem with this most basic build process. It's possible certain things have been fixed since those other examples were written. If you find you do need special build options you can simply add them as you normally would using configure.

```
cd /tmp
tar xjf /opt/mortoray.com/source-packages/freetype-2.4.10.tar.bz2
cd freetype-2.4.10/
./configure --host=arm-linux-androideabi --prefix=/opt/android-ext/
make
make install
```

The only catch with freetype is that the header files are put in the "freetype2" subdirectory. I suppose this is a historical oddity when it had to coexist with the previous freetype. This means you'll need to tell the compiler where to find the freetype headers (you will also have to do this for any Linux project using FreeType2). You can do this in several ways I believe, but I just stick the include path into my "Application.mk" file.

```
PLATFORM_PREFIX := /opt/android-ext/
APP_CPPFLAGS := -I$(PLATFORM_PREFIX)/include/freetype2/
```

Results

I use the above method and I have demos which load PNG images and render TrueType fonts on my Android device. Once I had everything working I tried it fresh again just to ensure it works as I intended. I can't guarantee it'll work on all build systems, but I hope at least it's a good start.

This work is part of my own effort to create a cross-platform framework for mobile game development. To speed up development I wanted to do most of my work strictly on a Linux machine. When building for Linux I simply use the Ubuntu versions of libraries like libpng. Once I have a bit more working I'll make that platform available.

Develop Power-Efficient Bluetooth Connected Wearables with Fast and Accurate Positioning.
LinkIt™ 2523 HDK by SAC
MEDIATEK labs

thoughts on "Android NDK Cross-Compile Setup (libpng and freetype)"

1. Pingback: hacklog, android lxc-tools | offlinehacker

2. Pingback: Cross-compiling a C library for Android NDK video

3. Steffen Pohle said:

   2013-01-27 at 22:46

   thanks for the good example on how to compile some librarys..

   REPLY

4. Pingback: Loading a PNG into Memory and Displaying It as a Texture with OpenGL ES 2, Using (Almost) the Same Code on iOS, Android, and Emscripten | Learn OpenGL ES

5. Koko said:

2013-10-30 at 14:02

Thanks for the libpng compile instructions. I missed a lot some guidelines on this out there.

REPLY

6. Sebastian *said:*

2014-06-09 at 09:23

Hello, I'm working for a FS i.MX6 based device(http://www.tbsdtv.com/launch/tbs-2910-matrix-arm-mini-pc.html)
I am trying to cross compile libCEC for Android, is there anybody here that have done this before for another Android device who can give me a hand or some lights?
Thanks

REPLY