May 28, 2016 by abhiemanyu | Android, Programming, Tutorial

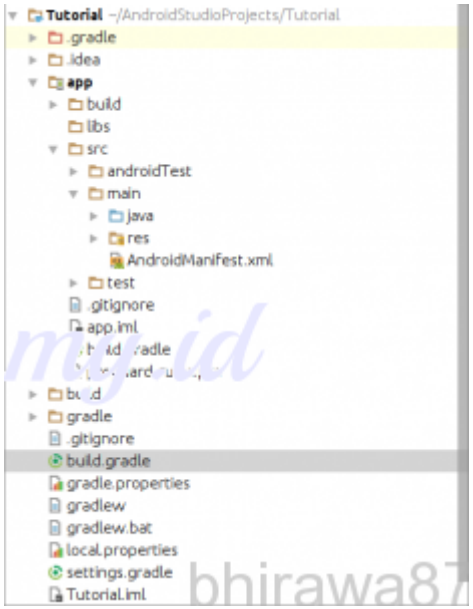
Android Studio CMake

Tweet

Android Studio 2.2 Preview 1 launched at Google I/O 2016. Android Studio allows you to develop with the latest Android APIs and features as mentioned here. UI layout editor is great but i really excited about **CMake** plugin. Using previous version, i was able to develop simple image scanner app that using NDK features. Now I'll try develop simple native app using CMake rather than ndk-build.

Create project, just like usual…but with cmake plugin

But remember to create project using Android Studio 2.2 Preview 1, you can download it here. If you already have previous version don't worry because you can install it side-by-side. When you're done, just create project like usual because the differences about enabling CMake features are when we use native code (C/C++).



That's typical folder structure in Android Studio. Next we create directory **cpp** inside **main** folder, and then create **CMakeLists.txt.** If you're not familiar with CMakeLists.txt find out more here and here.

```
1 cmake_minimum_required(VERSION 2.8.7)
2
3 add_library(hello-jni SHARED hello-jni.c)
4
5 target_link_libraries(hello-jni log android)
```

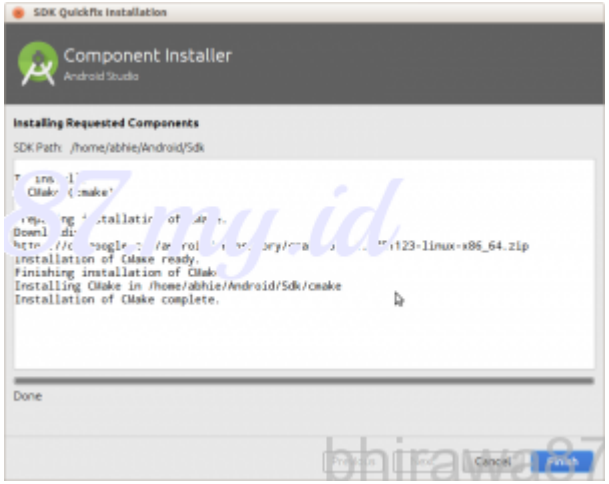
1. `cmake_minimum_required(VERSION 2.8.7)` means we need minimum version 2.8.7 of cmake. It depends on what you need in your project. by default Android Studio built in CMake version is 3.4.1
2. `add_library(hello-jni SHARED hello-jni.c)` means we create *shared library* named **hello-jni** with **hello-jni.c** as source.
3. `target_link_libraries(hello-jni log android)` means we link library needed by **hello-jni**, in this case **log** and **android**.\

Next create **hello-jni.c** as follow:

```
1  #include <string.h>
2  #include <jni.h>
3
4  jstring
5  Java_com_tutorial_MainActivity_stringFromJNI( JNIEnv* env,
6                                                jobject thiz )
7  {
8  #if defined(__arm__)
9    #if defined(__ARM_ARCH_7A__)
10   #if defined(__ARM_NEON__)
11     #if defined(__ARM_PCS_VFP)
12       #define ABI "armeabi-v7a/NEON (hard-float)"
13     #else
14       #define ABI "armeabi-v7a/NEON"
15     #endif
16   #else
17     #if defined(__ARM_PCS_VFP)
18       #define ABI "armeabi-v7a (hard-float)"
19     #else
20       #define ABI "armeabi-v7a"
21     #endif
22   #endif
23 #else
24   #define ABI "armeabi"
25 #endif
26 #elif defined(__i386__)
```

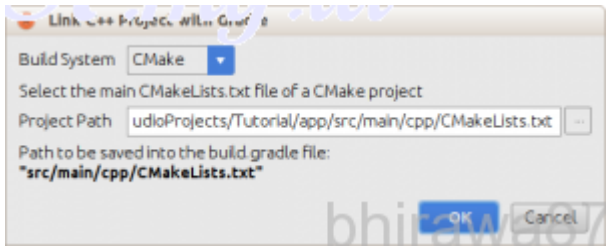
```
27    #define ABI "x86"
28  #elif defined(__x86_64__)
29    #define ABI "x86_64"
30  #elif defined(__mips64)  /* mips64el-* toolchain defines __mips__ too */
31    #define ABI "mips64"
32  #elif defined(__mips__)
33    #define ABI "mips"
34  #elif defined(__aarch64__)
35  #define ABI "arm64-v8a"
36  #else
37    #define ABI "unknown"
38  #endif
39
40    return (*env)->NewStringUTF(env, "Hello from JNI !  Compiled with ABI " ABI ".");
41  }
```

This code for checking ABI on device. Next right click on module name and then click "**Link C++ Project with Gradle**". For first time would be pop-up to install cmake plugin.



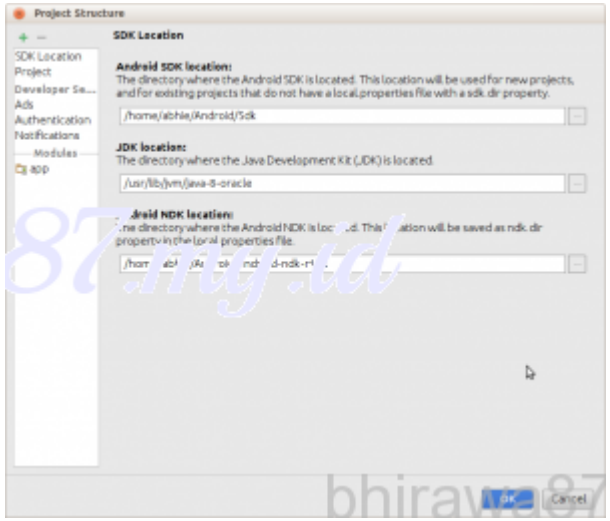
android studio cmake plugin installation

When installation done, point CMakeLists.txt to “Project Path”.



cmake project path

Since we still need NDK Framework to get cmake plugin works, make sure we already download NDK framework and set **ndk.dir** in *local.properties* or by **Project Structure >> SDK Location >>Android NDK Location.**



After syncing Gradle and no error reported, we can start invoking native method provide by **hello-jni.c** from activity. In this example print out ABI to textview. Here's my layout

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/activity_main"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context="com.tutorial.MainActivity">
9
10    <TextView
11       android:layout_width="wrap_content"
12       android:layout_height="wrap_content"
13       android:text="Hello World!"
14       app:layout_constraintBottom_toBottomOf="@+id/activity_main"
15       app:layout_constraintLeft_toLeftOf="@+id/activity_main"
16       app:layout_constraintRight_toRightOf="@+id/activity_main"
17       app:layout_constraintTop_toTopOf="@+id/activity_main"
18       android:id="@+id/textview"
19       android:textAppearance="@style/TextAppearance.AppCompat.Medium" />
```

```
20
21 </android.support.constraint.ConstraintLayout>
```

And here's my activity

```
1  public class MainActivity extends AppCompatActivity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.activity_main);
7
8          TextView hello = (TextView) findViewById(R.id.textview);
9
10         hello.setText(stringFromJNI()); //print out string returned by Native method
11
12     }
13
14     private native String stringFromJNI();
15
16     static {
17         System.loadLibrary("hello-jni");
18     }
19 }
```

Highlighted rows are invoking native method to activity, remember that we load library named **hello-jni** as we declared in CMakeLists.txt. Next try to build and run it to real device or emulator and should looks like this

I think it’s pretty straightforward to implementing cmake plugin with ndk rather than using Android.mk as earlier native build configuration because it gives us flexibility when using a lot of dependencies. But this is experimental anyway, until i post this, Android Studio 2.2 Preview 2 is available to download. So, i appreciate any suggestion by leaving comment below, see you next post.

[UPDATE July, 20th 2016]

Android Studio 2.2 Preview 6 available on canary channel. This update address bugs on Preview 5 see release notes for details. For me, i found bugs on Preview 5 that is

Android Studio no longer recognize cmake plugins, cannot resolve c/c++ and bad performance.

Gradle DSL method not found: ‘cmake()’ error solve

By updating gradle plugin to 2.2.0.alpha6 you have to change cmake config at

`defaultConfig`

```
defaultConfig {
      applicationId "com.example.abhie.skripsi"
      minSdkVersion 15
      targetSdkVersion 23
      ...

      cmake {
         abiFilters 'x86', 'x86_64', 'armeabi', 'armeabi-v7a'
      }
   }
```

to

```
defaultConfig {
      applicationId "mahoni.com.realtimecam"
      minSdkVersion 15
      targetSdkVersion 23
      ...
      ...
      externalNativeBuild {
         cmake {
//            targets "target1", "target2"
//            arguments "-DANDROID_TOOLCHAIN=clang"
//            cFlags "-DTEST_C_FLAG1", "-DTEST_C_FLAG2"
```

```
12  //            cppFlags "-DTEST_CPP_FLAG2", "-DTEST_CPP_FLAG2"
13  //            other configuration
14              abiFilters "armeabi-v7a", "armeabi"
15          }
16      }
17
18    }
```

remember to update gradle plugin to 2.2.0.alpha 6, there are two ways to update, first let android studio fix it by clicking “Fix and sync project” when error comes up on messages gradle build. or by updating root `build.gradle` (not module build.gradle)

```
1  dependencies {
2      classpath 'com.android.tools.build:gradle:2.2.0-alpha6'
3    }
```

abhiemanyu

← Perbedaan Bareksa.com dan Ipotfund • Google I/O 2016 →

8 thoughts on “Android Studio CMake”

Pingback: Android Studio CMake | Bhirawa Group's Blog

Olivier says:
June 16, 2016 at 10:01 pm

Great post.
Do you know how to select the abi to be generative from native code ?
By default, all ABI are generated (armv7, arm64, mips, x86, x86_64, mips64).
Do you know if it is possible to only build one ABI ?

Reply

abhiemanyu says:
July 12, 2016 at 8:21 am

you can add abiFilters cmake config at `defaultConfig` in build.gradle, like

```
defaultConfig{    cmake{      abiFilters 'x86','x86_64'    }}
```

Reply

abhiemanyu says:
July 20, 2016 at 1:19 pm

Hi Oliver,

It seems android studio 2.2 preview 6 available, on Preview 5 there are some bugs affect to cmake plugin. As i mentioned on previous reply that you need to add abiFilters at defaultConfig, this cause error if you update to preview 6 or even update LLDB (i'm not sure about LLDB). You can solve it by little edit

```
defaultConfig{   externalNativeBuild{      cmake{       abiFilters 'x86',       // some other config      }  }}
```

As you can see `externalNativeBuild` added. see this link

Reply

tony says:

October 6, 2016 at 10:15 pm

I use the following CMakelist.txt but my third party library isn't packaged into apk. so anyting wrong?

```
1  [crayon-5a0843faca984611285644 inline="true" ]<span class="pln">cmake_minimum_required</span><span cl
   ass="pun">(</span><span class="pln">VERSION </span><span class="lit">3.4</span><span class="pun">.</spa
   n><span class="lit">1</span><span class="pun">)</span><span class="pln">
2  include_directories</span><span class="pun">(</span><span class="pln">libs</span><span class="pun">/</sp
   an><span class="pln">arm</span><span class="pun">/</span><span class="pln">include</span><span
   class="pun">)</span><span class="pln">
3  link_directories</span><span class="pun">(</span><span class="pln">libs</span><span class="pun">/</span>
   <span class="pln">arm</span><span class="pun">/</span><span class="pln">lib</span><span class="pun">/)
   </span><span class="pln">
4
5  add_library</span><span class="pun">(</span> <span class="com"># Sets the name of the library.</span>
6          <span class="kwd">native</span><span class="pun">-</span><span class="pln">lib
7
8          </span><span class="com"># Sets the library as a shared library.</span><span class="pln">
9          SHARED
10
11         </span><span class="com"># Provides a relative path to your source file(s).</span>
12         <span class="com"># Associated headers in the same location as their source</span>
13         <span class="com"># file are automatically included.</span><span class="pln">
14         src</span><span class="pun">/</span><span class="pln">main</span><span class="pun">/</span><sp
   an class="pln">cpp</span><span class="pun">/</span><span class="kwd">native</span><span class="pun">-
   </span><span class="pln">lib</span><span class="pun">.</span><span class="pln">cpp </span><span
   class="pun">)</span><span class="pln">
15
16
17
```

```
find_library</span><span class="pun">(</span> <span class="com"># Sets the name of the path variable.</span><span class="pln">
        log</span><span class="pun">-</span><span class="pln">lib

        </span><span class="com"># Specifies the name of the NDK library that</span>
        <span class="com"># you want CMake to locate.</span><span class="pln">
        log </span><span class="pun">)</span><span class="pln">

target_link_libraries</span><span class="pun">(</span> <span class="com"># Specifies the target library.</span>
            <span class="kwd">native</span><span class="pun">-</span><span class="pln">lib

            </span><span class="com"># Links the target library to the log library</span>
            <span class="com"># included in the NDK.</span><span class="pln">
            $</span><span class="pun">{</span><span class="pln">log</span><span class="pun">-</span><span class="pln">lib</span><span class="pun">}</span> <span class="pun">)</span><span class="pln">
add_library</span><span class="pun">(</span><span class="pln">avcodec</span><span class="pun">-</span><span class="lit">57</span><span class="pln"> SHARED IMPORTED</span><span class="pun">)</span><span class="pln">
set_target_properties</span><span class="pun">(</span><span class="pln">avcodec</span><span class="pun">-</span><span class="lit">57</span><span class="pln"> PROPERTIES IMPORTED_LOCATION C</span><span class="pun">:</span><span class="str">/Users/</span><span class="pln">tony</span><span class="pun">/</span><span class="typ">Desktop</span><span class="pun">/</span><span class="typ">MyApplication</span><span class="pun">/</span><span class="pln">app</span><span class="pun">/</span><span class="pln">libs</span><span class="pun">/</span><span class="pln">arm</span><span class="pun">/</span><span class="pln">lib</span><span class="pun">/</span><span class="pln">libavcodec</span><span class="pun">-</span><span class="lit">57.so</span><span class="pun">)</span><span class="pln">
target_link_libraries</span><span class="pun">(</span><span class="kwd">native</span><span class="pun">-</span><span class="pln">lib avcodec</span><span class="pun">-</span><span class="lit">57</span><span class="pun">)</span><span class="pln">
target_link_libraries</span><span class="pun">(</span><span class="kwd">native</span><span class="pun">-</span><span class="pln">lib avformat</span><span class="pun">-</span><span class="lit">57</span><span class="pun">)</span><span class="pln">

target_link_libraries</span><span class="pun">(</span><span class="kwd">native</span><span class="pun">-</span><span class="pln">lib avutil</span><span class="pun">-</span><span class="lit">55</span><span class="pun">)</span><span class="pln">
target_link_libraries</span><span class="pun">(</span><span class="kwd">native</span><span class="pun">-</span><span class="pln">lib avfilter</span><span class="pun">-</span><span class="lit">6</span><span class="pun">)</span>
```

[/crayon]

Reply

abhiemanyu says:
October 11, 2016 at 12:32 pm

hi Tony,

In order to use third party libs, you should include it by `find_libray(<lib_name>)`, but before cmake can “find” the libs, you should include your third party `INCLUDE_DIR` by calling `include_directories(<dir_path>)`.

include_directories docs

hope this helps

Reply

starinline says:

December 3, 2016 at 1:39 pm

Canot find “**Link C++ Project with Gradle**”，Why？

Reply

abhiemanyu says:

December 8, 2016 at 1:03 pm

What version of Android Studio do you use? Use 2.2 or latest

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Visual Text

Name *

Email *

Website

Post Comment