Running CMake from command line within an Android NDK project

How might one run an NDK Cmake build independently from the rest of an Android project, ideally from the command line, external to Android Studio?

The equivalent of running `ndk-build` from the jni directory for slightly older Android NDK projects.

I need to investigate exactly what the calls to the compiler look like, and I can't seem to get this information when building the whole project from within Android Studio

My first attempt was just to run cmake from the `project/app` directory containing `CMakeLists.txt`, but this informs me that `cmake` is not installed - so how is Android Studio managing to build it then?

android android-ndk cmake

edited Jun 21 '17 at 9:51

asked Jun 21 '17 at 9:42

bph
3,627 5 37 78

ok - have now found the Android/Sdk/cmake/3.6.3155560/bin directory - making progress.. – bph Jun 21 '17 at 9:58

This is a relevant and useful resource -> developer.android.com/ndk/guides/cmake.html#build-command – bph Jun 21 '17 at 11:11

This may be relevant to what you're trying to do, although it's not the same as your question: stackoverflow.com/questions/46530158/… – James Moore Nov 14 '17 at 20:16

If your goal is to just run from the command line (as opposed to trying to do *exactly* what gradle is doing), just use cmake the way you normally would:

```
$ cmake -DCMAKE_TOOLCHAIN_FILE=$NDK/build/cmake/android.toolchain.cmake \
    -DANDROID_ABI=whatever $YOUR_SOURCE_DIR
```

Alternatively, you can just run `./gradlew` from the command line.

answered Jun 22 '17 at 19:11

Dan Albert
3,467 1 13 37

1 this is correct, but I discovered that the way to inspect the clang compiler calls was to step one level deeper and run the build.ninja files with the -v option – bph Jun 22 '17 at 20:31

`-DANDROID_PLATFORM=android-XXX` may also be necessary, where XXX is 21, or 16, or whatever you need. – Alex Cohn Aug 7 '17 at 8:19

Your original problem is that you cannot see the command-line invocation when building with Android Studio.

You can get the command line arguments to the compiler by editing your app/build.gradle file.

```
defaultConfig {
    ...
    externalNativeBuild {
        cmake {
            ...
            arguments "-DCMAKE_VERBOSE_MAKEFILE=1", ...
        }
    }

}
```

like 30.

[1/176] /home/bram/android-sdk-linux/ndk-bundle/toolchains/llvm/prebuilt/linux-x86_64/bin/clang --target=armv7-none-linux-androideabi --gcc-toolchain=/home/bram/android-sdk-linux/ndk-bundle/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64 --sysroot=/home/bram/android-sdk-linux/ndk-bundle/sysroot -isystem /home/bram/android-sdk-linux/ndk-bundle/sysroot/usr/include/arm-linux-androideabi -D__ANDROID_API__=19 -g -DANDROID -ffunction-sections -funwind-tables -fstack-protector-strong -no-canonical-prefixes -march=armv7-a -mfloat-abi=softfp -mfpu=vfpv3-d16 -fno-integrated-as -marm -mfpu=neon -Wa,--noexecstack -Wformat -Werror=format-security -Os -DNDEBUG -fPIC -MD -MT /home/bram/src/GPGOAP/CMakeFiles/gpgoap.dir/astar.c.o -MF /home/bram/src/GPGOAP/CMakeFiles/gpgoap.dir/astar.c.o.d -o /home/bram/src/GPGOAP/CMakeFiles/gpgoap.dir/astar.c.o -c /home/bram/src/GPGOAP/astar.c

answered Jun 22 '17 at 21:38

Bram

2,006 16 39

this is interesting as I did add set(CMAKE_VERBOSE_MAKEFILE on) to my CMakeLists.txt instead of the build.gradle file but it didn't seem to do anything. I assume that both approaches should effectively achieve the same end? I'll go back and double check.. – bph Jun 22 '17 at 21:49