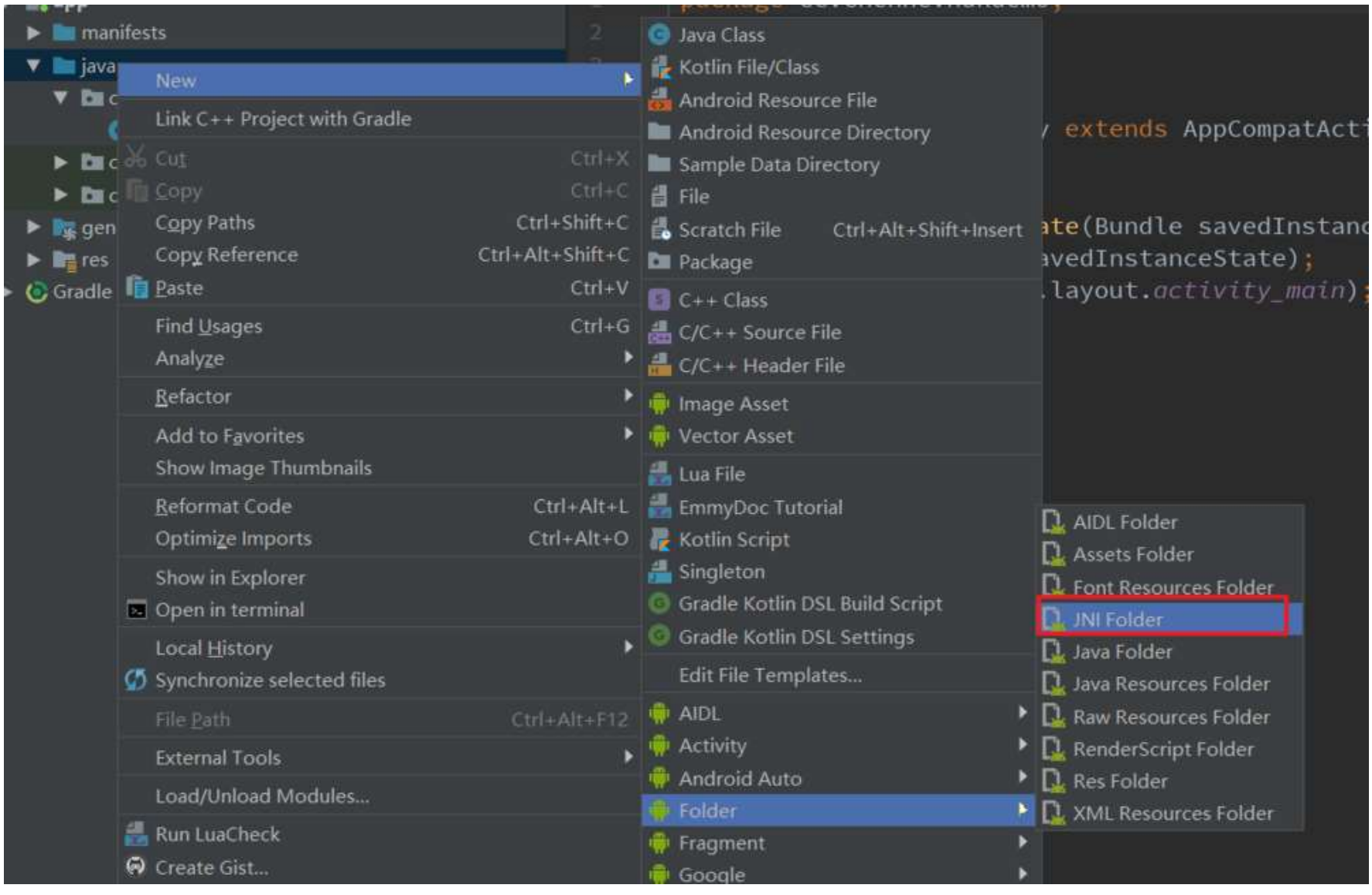
Android Studio 3.3 **ndk** 环境搭建

Android Studio 嵌入 C 代码的过程，使用 ndk-build. 当前环境：Android 研发群 647986362

- Android Studio 3.3
- NDK 18.1

创建 JNI 文件夹

直接在项目右键，选择 New - Folder - JNI Folder ，对话框直接点击 Finish 即可方便地在默认位置创建 jni 文件夹用于存放 c 源码。默认位置在 app/src/main/jni.



创建 Java 类

首先创建一个 Java 类用于调用 c 代码。

```
public class JniTest {

    static {
        System.loadLibrary("JniLib");
    }

    public static native String getString();
}
```

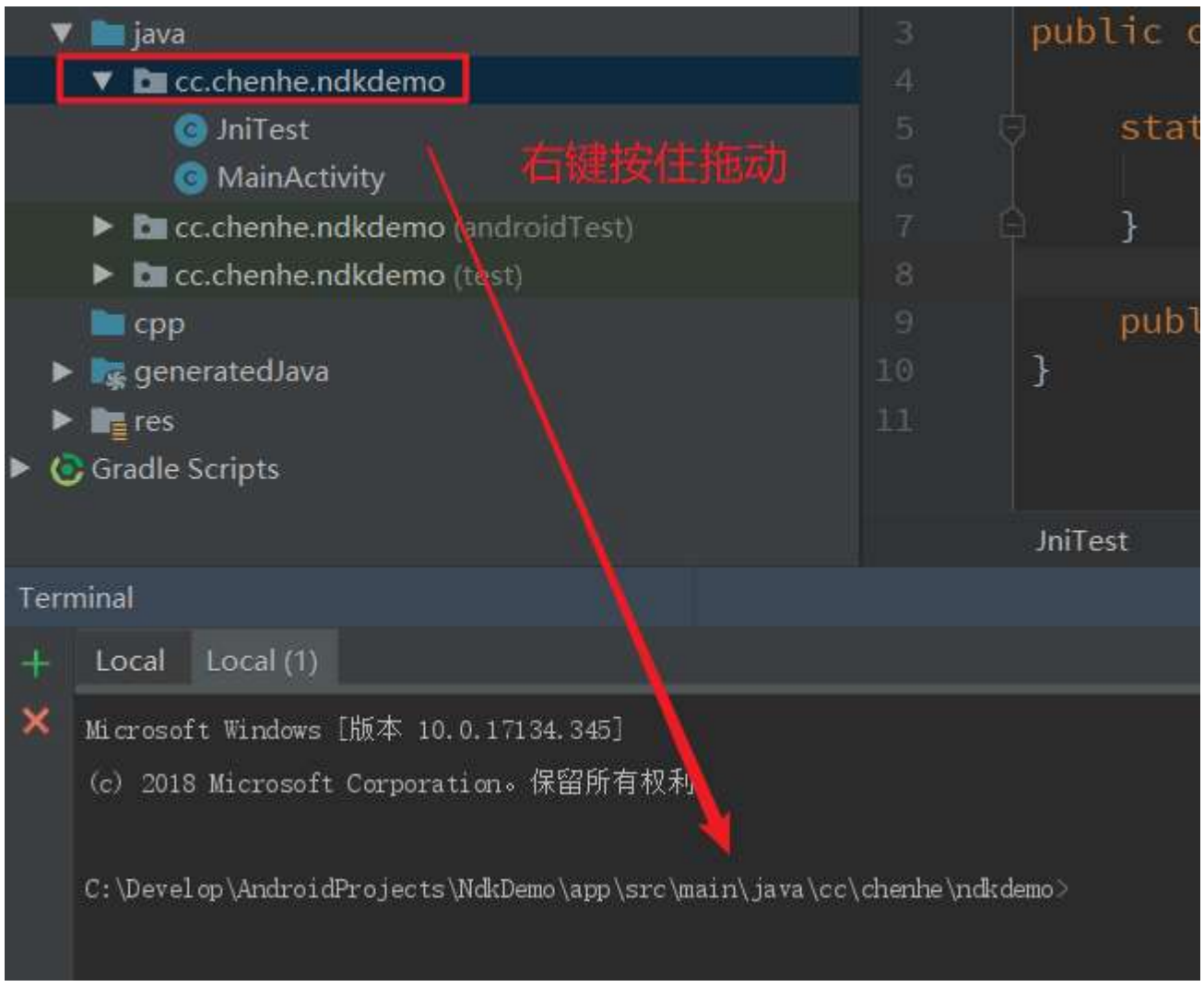
生成头文件 (.h)

命令行

最直接的方式就是通过命令行生成。

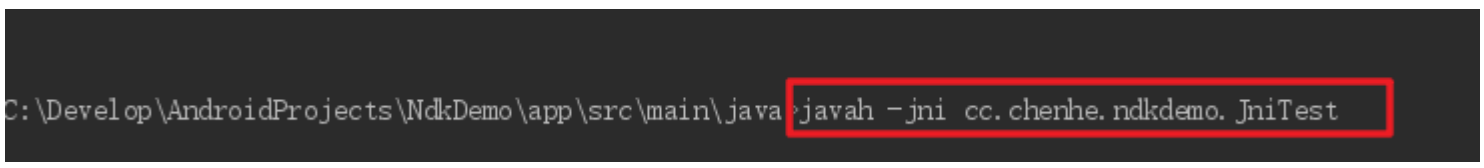
首先使用 javac 编译 java 文件。

小技巧。使用右键按住拖动文件夹到终端面板，可以快速进入对应目录。

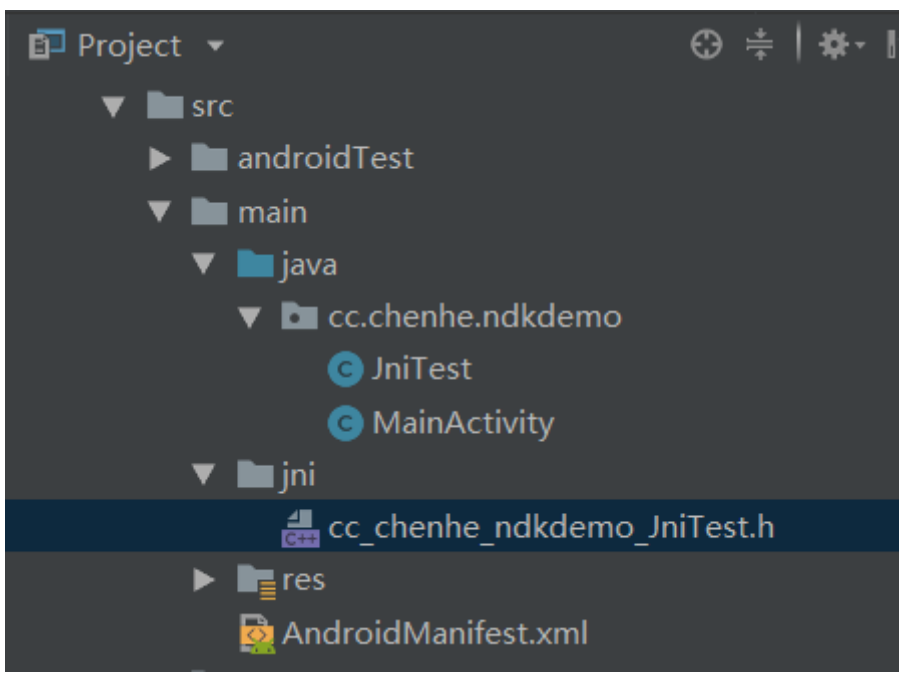


进入到文件所在目录后执行 javac JniTest.java 编译。成功后会出现 JniTest.class 文件。

然后**退回到包外目录**执行 javah -jni 生成头文件。



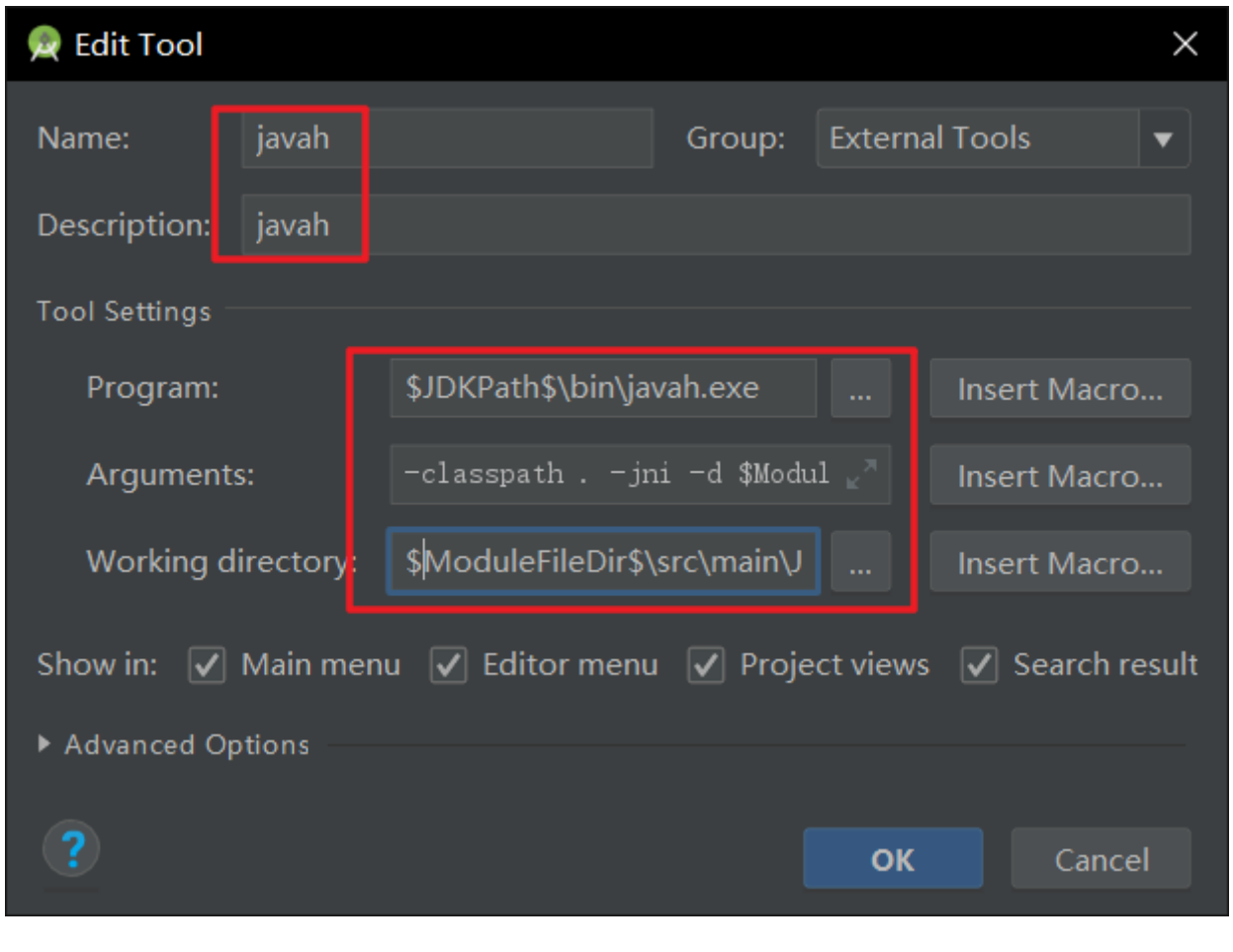
注意径不要写错了，最后也不需要加文件扩展名。成功后会生成一个 .h 文件，把它手动移到 jni 目录。之前编译出的 .class 文件可以删掉了。最终目录结构如下：



配置外部工具

每次都这样搞一遍很麻烦，我们可以配置一下扩展工具，这样一劳永逸。

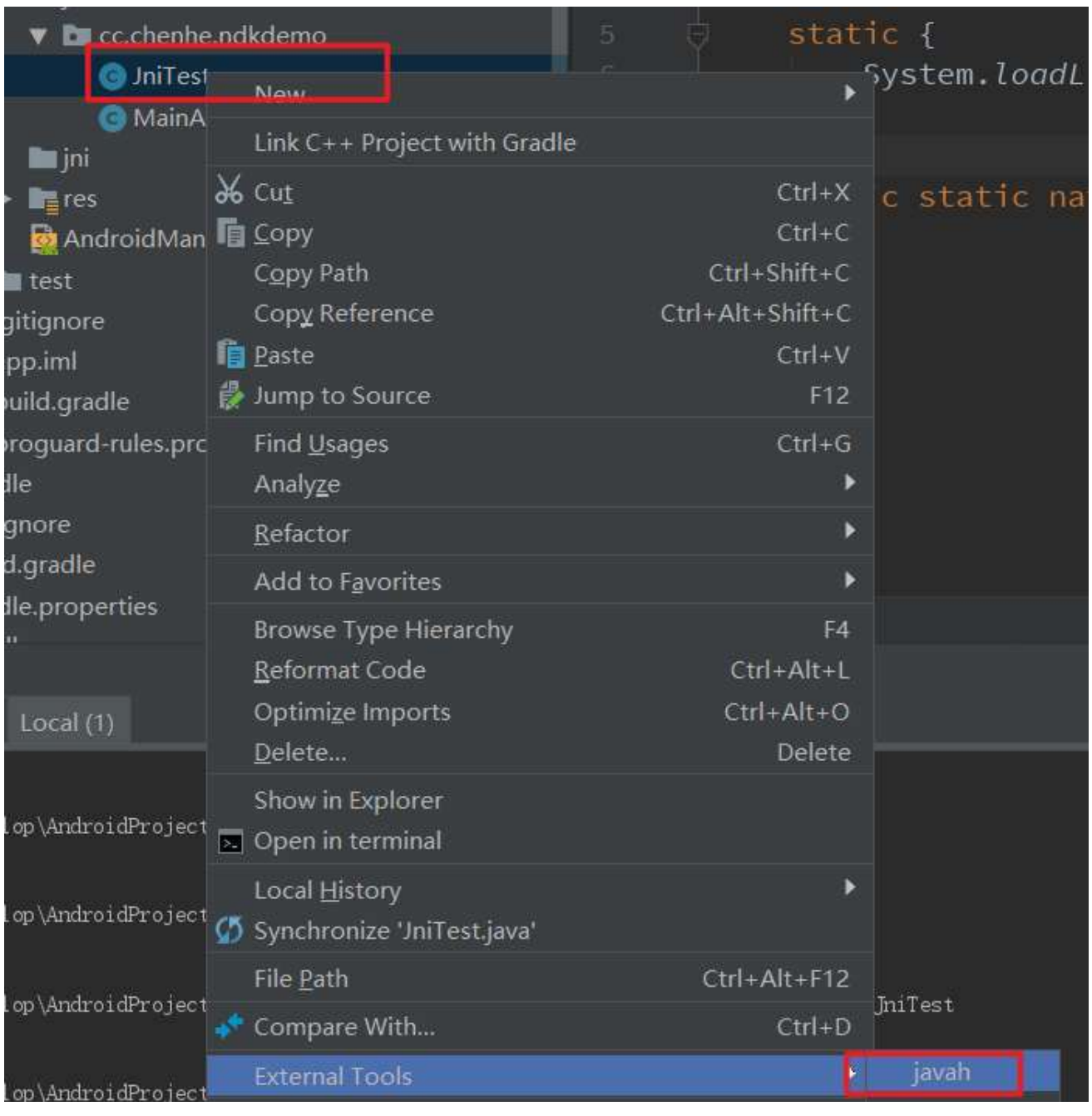
点击 File - Setting - Tools - External Tools 打开外部工具配置页，点击 + 新建一个工具。



先起个名字叫，这里叫做 javah.

- Program: $JDKPath$\bin\javah.exe
- Arguments: -classpath . -jni -d $ModuleFileDir$\src\main\jni $FileClass$
- Working directory: $ModuleFileDir$\src\main\Java

点击 OK 保存后就新建了一个工具。此时我们右击 JniTest.java，在菜单中选择 External Tools - javah 就可以快速生成头文件并放到 jni 目录。



编写 c 代码

在 jni 目录新建 一个 c 语言源码，这里叫做 JinLib.cpp. 然后实现头文件中所定义的函数，别忘引入头文件。这里简单地返回一个字符串：

```
// 头文件 可能不同
#include  <cc_chenhe_ndkdemo_JniTest.h>
/*
 * Class:     cc_chenhe_ndkdemo_JniTest
 * Method:    getString * Signature: ()Ljava/lang/String;
 */
 JNIEXPORT jstring JNICALL Java_cc_chenhe_ndkdemo_JniTest_getString
  (JNIEnv * env, jclass){
    return (*env).NewStringUTF("Hello cpp");
  }
```

创建 mk 文件

mk 文件用于告诉 ndk-build 该如何编译 c 源码，详情见官方指南。

在 jni 目录下创建 Android.mk：

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := JniLib
LOCAL_SRC_FILES =: JniLib.cpp
include $(BUILD_SHARED_LIBRARY)
```

其中 LOCAL_SRC_FILES 列出了所有要编译的 c 源码文件。

然后创建 Application.mk：

```
APP_MODULES := JniLib
APP_ABI := all
```

gradle 配置

在 module 的 build.gradle 里，amndroid.defaultConfig 下加入下面配置：

```
ndk{
    moduleName "JniLib"
    //abiFilters \"armeabi-v7a", "x86" //输出指定 abi 下的 so 库
}
sourceSets.main{
    jni.srcDirs = []
    jniLibs.srcDir "src/main/libs"
}
```

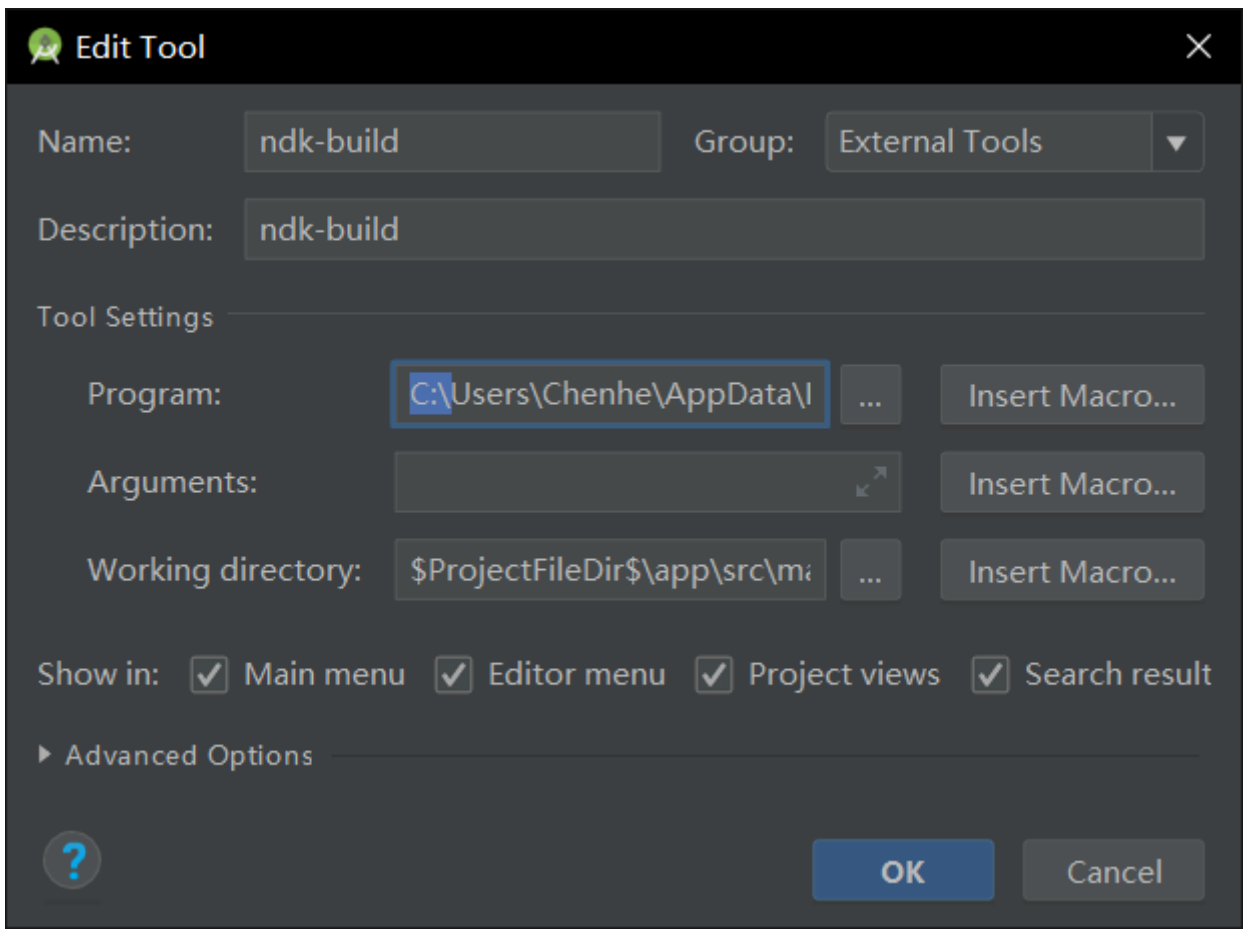
```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cc.chenhe.ndkdemo"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

        ndk{
            moduleName "JniLib"
//            abiFilters "armeabi", "armeabi-v7a", "x86" //输出指定的三种abi体系下的so库
        }
        sourceSets.main{
            jni.srcDirs = []
            jniLibs.srcDir "src/main/libs"
        }

    }
```

编译

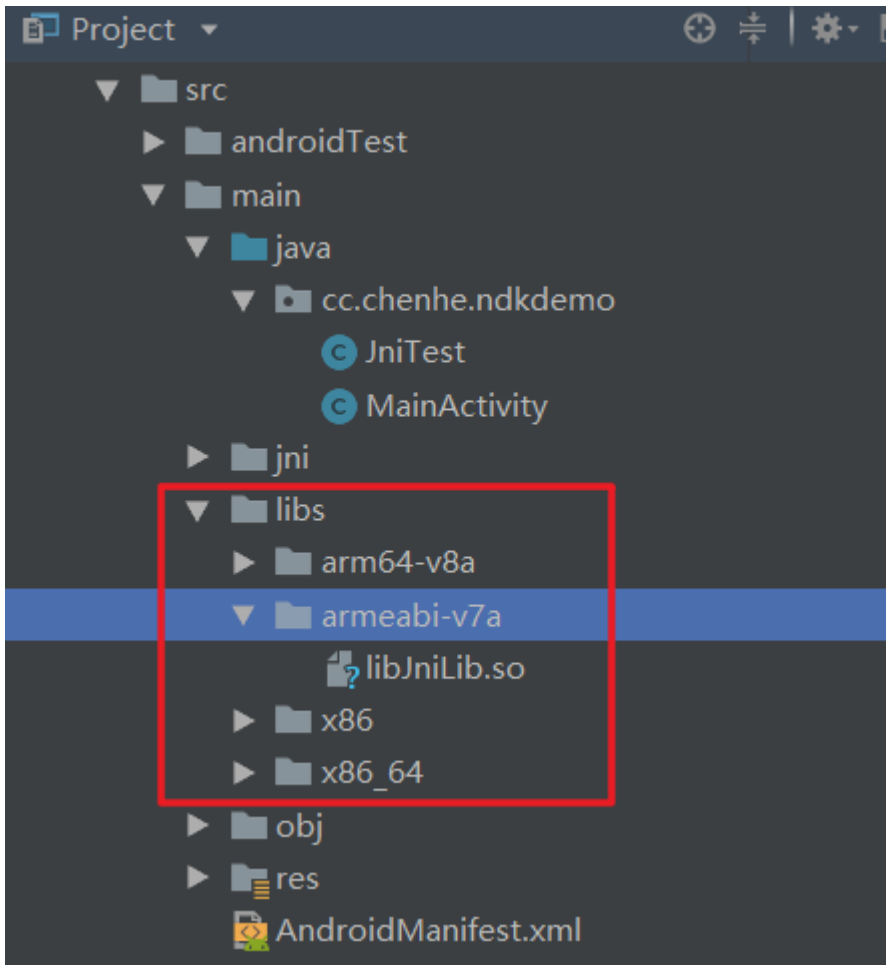
编译需要使用 ndk-build，其位于 ndk 目录下。如果要直接在命令行使用需要添加环境变量。类似的，为了方便我也添加了一个外部工具。



- Program: C:\Users\Chenhe\AppData\Local\Android\Sdk\ndk-bundle\build\ndk-build.cmd
- Working directory: $ProjectFileDir$\app\src\main

注意改成你自己的目录。

任意找个第地方右击，选择 External Tools - ndk-build 即可编译 c 源码。成功后可以看见创建了 libs 目录，里面包含了不同平台下的 so 文件。



运行

最后修改下 Activity 的代码：

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView tv = findViewById(R.id.text_view);
        tv.setText(JniTest.getString());
    }
}
```

运行后就可以看到效果了：

来源：Android 研发群 647986362