

# 前进的小巨人

博客园 首页 新随笔 联系 管理 订阅 XML

随笔- 35 文章- 5 评论- 15

昵称：[Alif](#)  
园龄：[2年6个月](#)  
粉丝：[12](#)  
关注：[7](#)  
[+加关注](#)

## Android Studio使用JNI

### 0x01 前言

本文讲述使用Android Studio通过静态注册、动态注册使用JNI的方法，以及加载第三方so文件的方法

### 0x02 Android Studio静态注册的方式使用JNI

#### 1. 添加native接口



```
public class MainActivity extends Activity implements OnClickListener {  
  
    static{  
        System.loadLibrary("JniTest");  
    }  
  
    private native int Add(double num1,double num2);  
    private native int Sub(double num1,double num2);  
    private native int Mul(double num1,double num2);  
    private native int Div(double num1,double num2);  
}
```

2017年12月						
<	日	一	二	三	四	五
	26	27	28	29	30	1
	3	4	5	6	7	8
	10	11	12	13	14	15
	17	18	19	20	21	22
	24	25	26	27	28	29
	31	1	2	3	4	5


## 搜索

## 常用链接

[我的随笔](#)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
}
}
```



在Java类中使用System.loadLibrary("JniTest")加载我们要写的so库名称，Add/Sub/Mul/Div这四个方法在Java类中声明就可以使用了。

## 2.Build->Make Project

验证工程中并无其他错误，并对工程进行编译，生成.class文件  
在Build/intermediates/classes/debug里面

## 3.javah 生成.h文件

cmd 进入工程目录在工程的app/src/main/java，执行 javah com.example.calculate.MainActivity 命令，就会生成so库所需要的.h文件

在/app/src/main/下建立jni目录，将.h拷贝进去

## 4.jni目录下新建一个.c文件，完成so库的函数实现(注:这里的Android.mk文件并不需要，可以不写，用法在下面会提到)

[我的评论](#)  
[我的参与](#)  
[最新评论](#)  
[我的标签](#)

## 我的标签

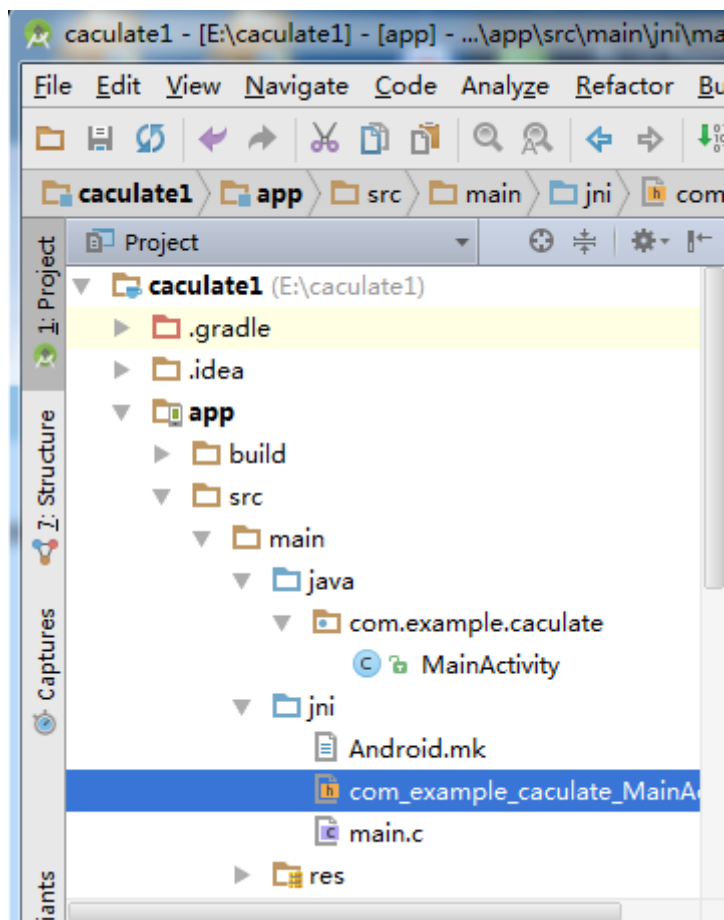
[64位函数栈\(1\)](#)  
[64位栈\(1\)](#)  
[64位栈帧\(1\)](#)  
[cuckoo\(1\)](#)  
[IntegrityError\(1\)](#)

## 随笔分类

[c++\(1\)](#)  
[Linux](#)  
[python](#)  
[vt](#)  
[windows机制\(2\)](#)  
[windows漏洞挖掘](#)  
[windows驱动开发\(2\)](#)  
[安卓\(6\)](#)  
[函数\(2\)](#)  
[逆向分析\(3\)](#)  
[日常错误\(2\)](#)  
[软件调试\(6\)](#)  
[网络\(1\)](#)  
[主动防御\(2\)](#)

## 随笔档案

[2017年12月 \(1\)](#)  
[2017年11月 \(2\)](#)  
[2017年9月 \(1\)](#)  
[2017年8月 \(4\)](#)  
[2017年5月 \(1\)](#)  
[2016年12月 \(1\)](#)



5. Build->Make Project 会报错，这里在android studio中添加ndk路径编译生成so文件

①在local.properties文件中增加ndk的路径

2016年9月 (5)

2016年8月 (7)

2016年5月 (5)

2016年4月 (4)

2016年3月 (4)

## 最新评论

1. Re:ELF文件解析器支持x86x64ELF文件  
怎么使用?怎么打开文件?我的邮箱994825354@qq.com

--justtsuj

2. Re:安卓加固之so文件加固  
@y449756770文末有代码链接，可以参考一下当时是参考学习的，这个大牛的链接在开始也给了...

--Alif

3. Re:安卓加固之so文件加固  
楼主，第一种加密节的方法，加密的代码在哪里运行啊？怎么运行？光看到代码不知道怎么运行，我qq 449756770，能否指教一二，比较急

--y449756770

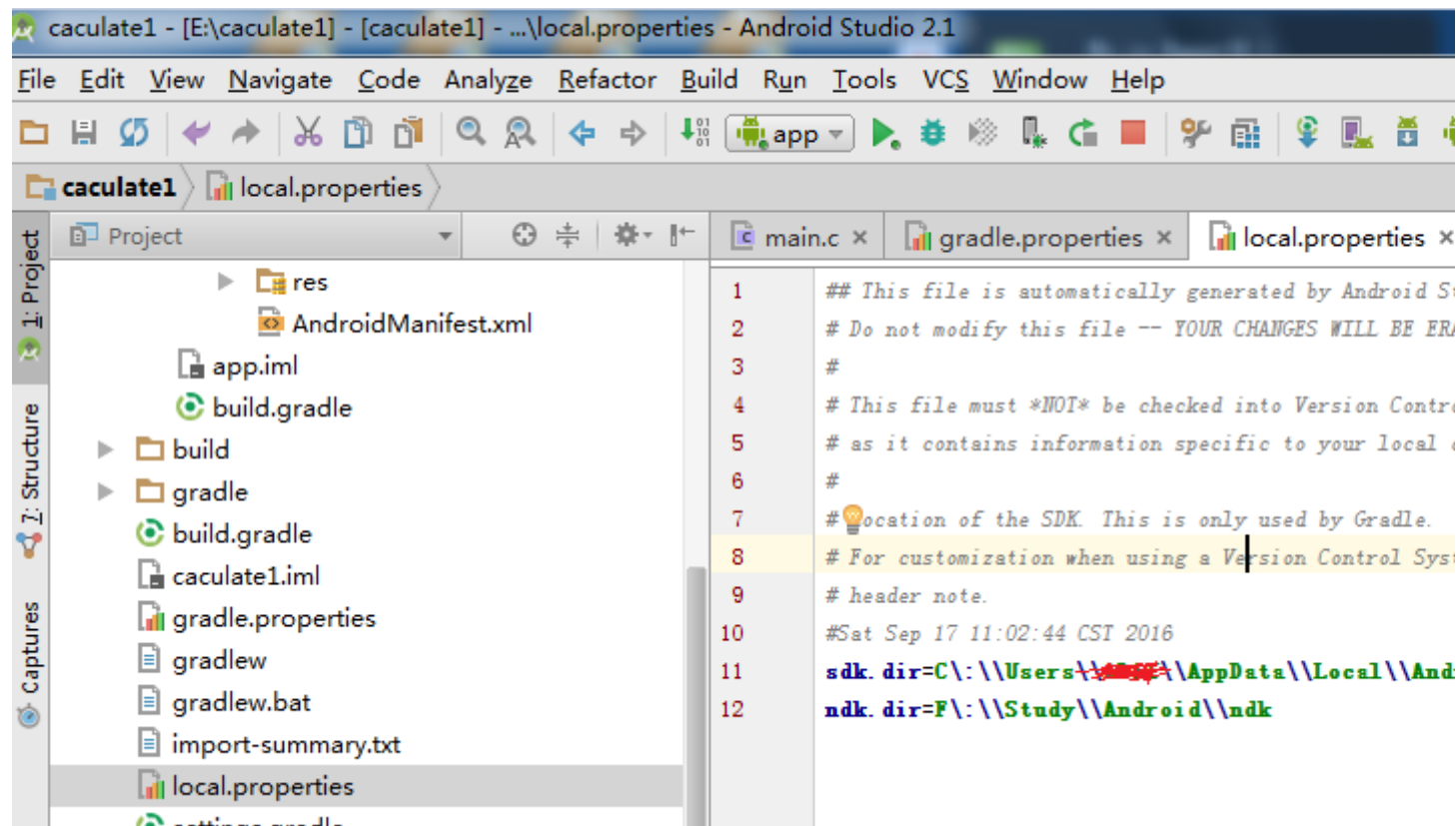
4. Re:Windows x64 栈帧结构  
@稻草人.cn二、三、四例子中都有三个局部变量，如果开辟的栈是一样的，那么都应该是sub rsp, 0x10; 但是只有例二是这样的，因为它的Add函数没有在调用其他函数。而例三中Sub函数调用了.....

--Alif

5. Re:Windows x64 栈帧结构  
000000013F931049 mov r9d,4 000000013F93104F mov r8d,3 000000013F931055 mov edx,2 000000013F93105A mo.....

--稻草人.cn

## 阅读排行榜



1. Android Studio使用JNI(6280)
2. Android下so注入和hook(4245)
3. Win7 x64下进程保护与文件保护(ObRegisterCallbacks)(3412)
4. IntegrityError错误(2615)
5. 安卓加固之so文件加固(1753)

## 评论排行榜

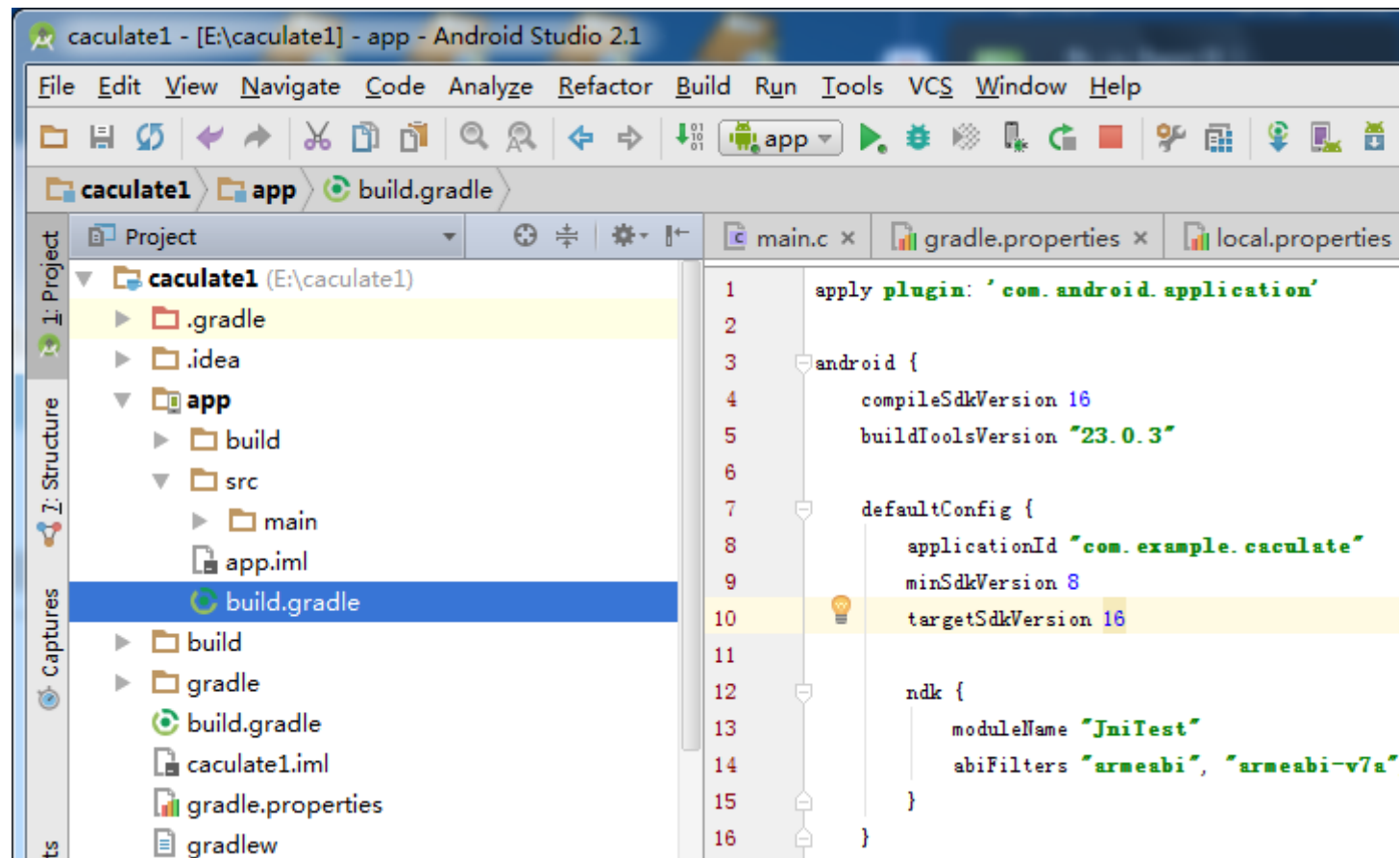
1. Windows x64 栈帧结构(5)
2. Android下so注入和hook(3)
3. Android Studio使用JNI(3)
4. 安卓加固之so文件加固(2)
5. ELF文件解析器支持x86x64ELF文件(1)

## 推荐排行榜

1. Windows x64 栈帧结构(4)

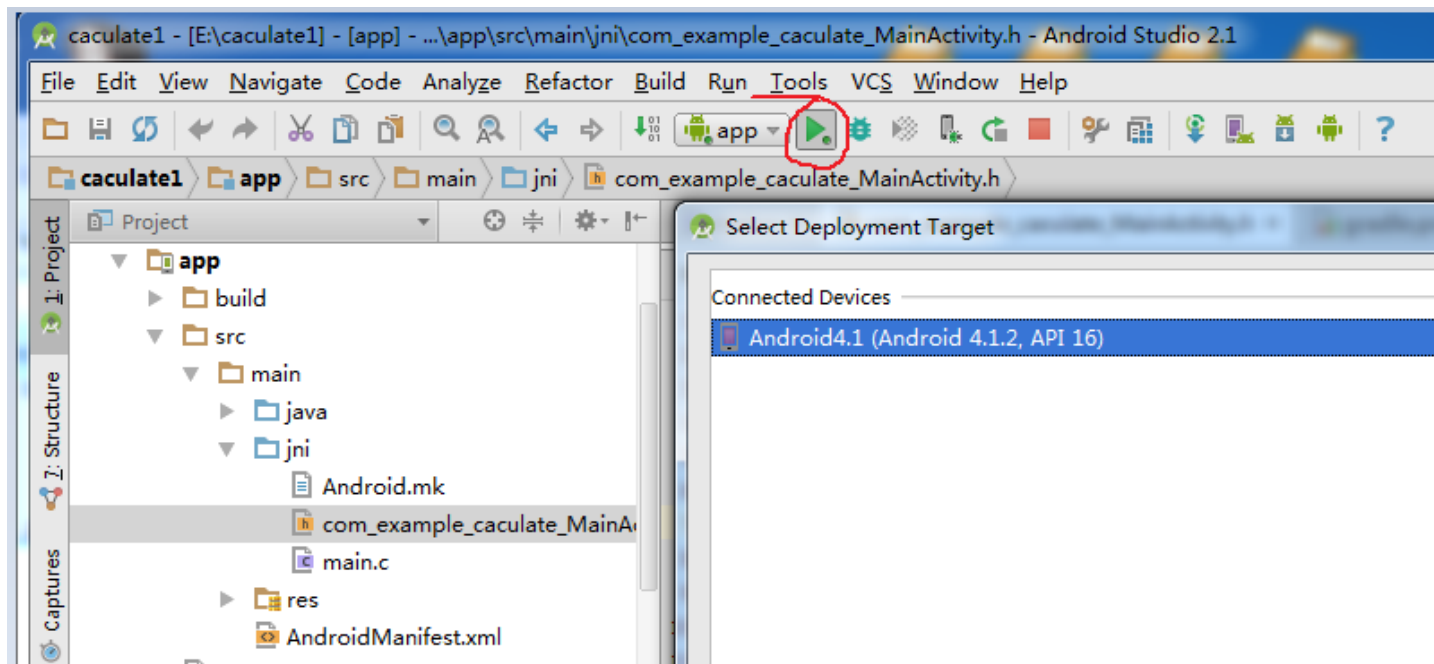
②在/app/目录下的build.gradle文件里增加 so的名称

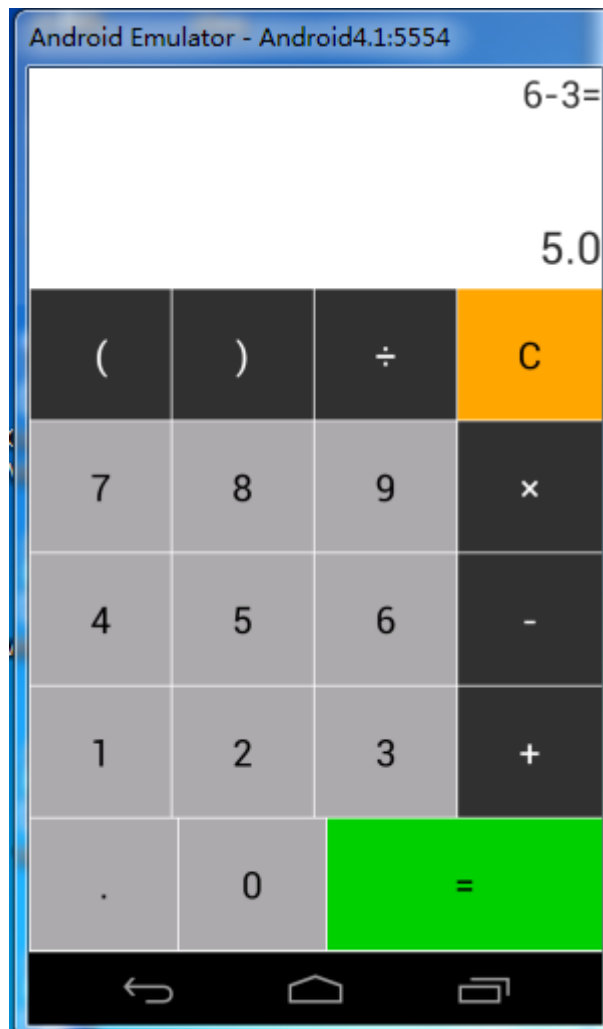
```
ndk {  
    moduleName "JniTest"  
    abiFilters "armeabi", "armeabi-v7a", "x86"  
}
```



6.Build->Make Project 就可以编译出so文件了  
生成的so在app/Build/intermediates/ndk/debug/lib下面

7.点击运行，就可以使用我们实现的so中的代码了





(注:在so库中实现在原结果基础上加2)

使用模拟器利用busybox安装grep命令之后，我们可以通过/proc/<pid>/maps文件中保存的进程加载模块，查看我们写的so文件是否被加载了。

```
1!root@android:/data/busybox # cat /proc/29466/maps|grep libJniTest.so
5a953000-5a955000 r-xp 00000000 1f:01 906      /data/data/com.example.caculate
/lib/libJniTest.so
5a955000-5a956000 r--p 00001000 1f:01 906      /data/data/com.example.caculate
/lib/libJniTest.so
5a956000-5a957000 rw-p 00002000 1f:01 906      /data/data/com.example.caculate
/lib/libJniTest.so
root@android:/data/busybox #
```

## 8.我们JNI的实现文件

### ①com\_example\_caculate\_MainActivity.h文件



```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_example_caculate_MainActivity */

#ifndef _Included_com_example_caculate_MainActivity
#define _Included_com_example_caculate_MainActivity
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      com_example_caculate_MainActivity
 * Method:     Add
 * Signature:  (DD)I
 */
JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Add
    (JNIEnv *, jobject, jdouble, jdouble);

/*
 * Class:      com_example_caculate_MainActivity
 * Method:     Sub
 * Signature:  (DD)I
 */
JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Sub
    (JNIEnv *, jobject, jdouble, jdouble);
```



```

/*
 * Class:      com_example_caculate_MainActivity
 * Method:     Mul
 * Signature:  (DD)I
 */
JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Mul
    (JNIEnv *, jobject, jdouble, jdouble);

/*
 * Class:      com_example_caculate_MainActivity
 * Method:     Div
 * Signature:  (DD)I
 */
JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Div
    (JNIEnv *, jobject, jdouble, jdouble);

#ifdef __cplusplus
}
#endif
#endif

```



## ②so库中实现函数的main.c文件



```

#include <jni.h>
#define jintJNICALL
#ifndef _Included_com_example_caculate_MainActivity
#define _Included_com_example_caculate_MainActivity
#ifdef __cplusplus
extern "C" {
#endif

JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Add
    (JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 + num2+2);
}

```

```
}

JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Sub
    (JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 - num2+2);
}

JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Mul
    (JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 * num2+2);
}

JNIEXPORT jint JNICALL Java_com_example_caculate_MainActivity_Div
    (JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    if (num2 == 0) return 0;
    return (jint)(num1 / num2+2);
}
#ifdef __cplusplus
}
#endif
#endif
```



### 0x03 Android Studio动态注册的方式使用JNI

1.jni目录下直接编写so库中的.c文件

JNI\_OnLoad会作为so库被加载后的第一个执行函数，最后通过RegisterNatives函数将JNI函数注册



```
#include <jni.h>
#include <stdio.h>
```

```
//#include <assert.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

JNIEXPORT jint JNICALL native_Add
(JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 + num2 +1);
}

JNIEXPORT jint JNICALL native_Sub
(JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 - num2 +1);
}

JNIEXPORT jint JNICALL native_Mul
(JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    return (jint)(num1 * num2 +1);
}

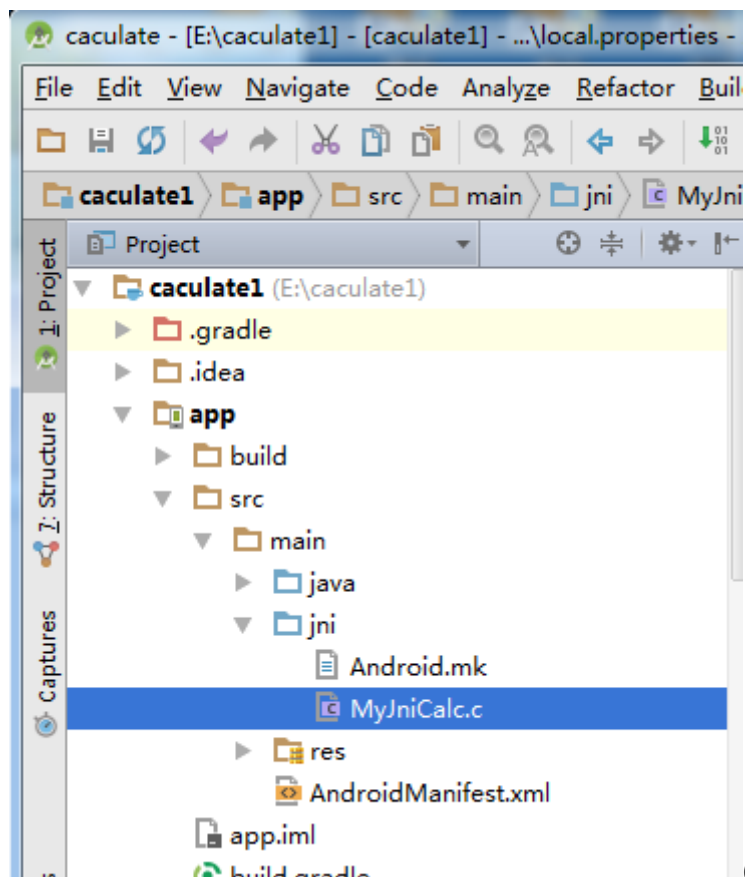
JNIEXPORT jint JNICALL native_Div
(JNIEnv *env, jobject obj, jdouble num1, jdouble num2)
{
    if (num2 == 0) return 0;
    return (jint)(num1 / num2 +1);
}

//Java和JNI函数的绑定表
static JNINativeMethod gMethods[] = {
    {"Add", "(DD)I", (void *)native_Add},
    {"Sub", "(DD)I", (void *)native_Sub},
```

```
    {"Mul", "(DD)I", (void *)native_Mul},  
    {"Div", "(DD)I", (void *)native_Div},  
};  
  
//注册native方法到java中  
static int registerNativeMethods(JNIEnv* env, const char* className,  
                                JNINativeMethod* gMethods, int numMethods)  
{  
    jclass clazz;  
    clazz = (*env)->FindClass(env, className);  
    if (clazz == NULL) {  
        return JNI_FALSE;  
    }  
    if ((*env)->RegisterNatives(env, clazz, gMethods, numMethods) < 0){  
        return JNI_FALSE;  
    }  
  
    return JNI_TRUE;  
}  
  
int register_ndk_load(JNIEnv *env)  
{  
  
    return registerNativeMethods(env, "com/example/caculate/MainActivity",  
                                gMethods, sizeof(gMethods) / sizeof(gMethods[0]));  
    //NELEM(gMethods));  
}  
  
JNIEXPORT jint JNI_OnLoad(JavaVM* vm, void* reserved)  
{  
    JNIEnv* env = NULL;  
    jint result = -1;  

```

```
if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_4) != JNI_OK) {  
    return result;  
}  
  
register_ndk_load(env);  
  
// 返回jni的版本  
return JNI_VERSION_1_4;  
}
```



(注:这里Android.mk文件也不需要,在下面会提到)

2.在Java类中添加native接口,加载JniTest.so,以及声明native函数,声明之后,函数可以直接使用。



```
public class MainActivity extends Activity implements OnClickListener {

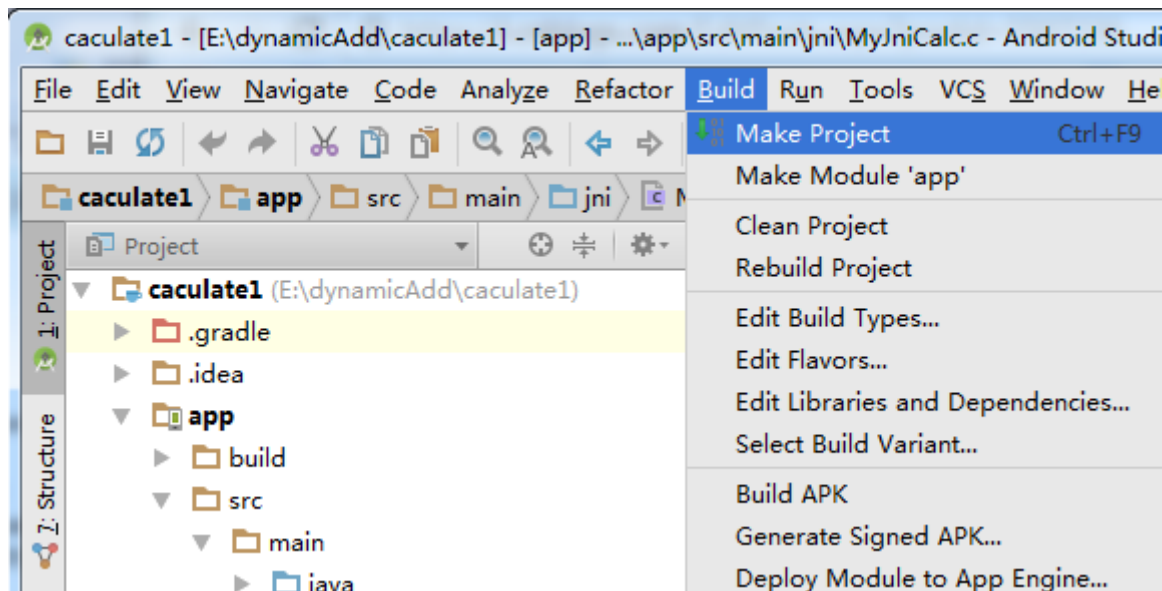
    static{
        System.loadLibrary("JniTest");
    }

    private native int Add(double num1,double num2);
    private native int Sub(double num1,double num2);
    private native int Mul(double num1,double num2);
    private native int Div(double num1,double num2);

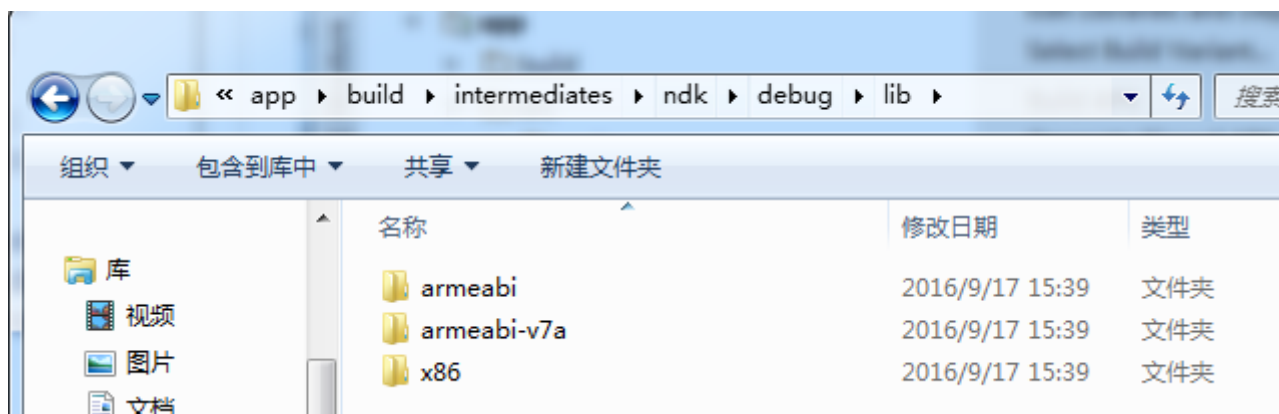
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    }
}
```



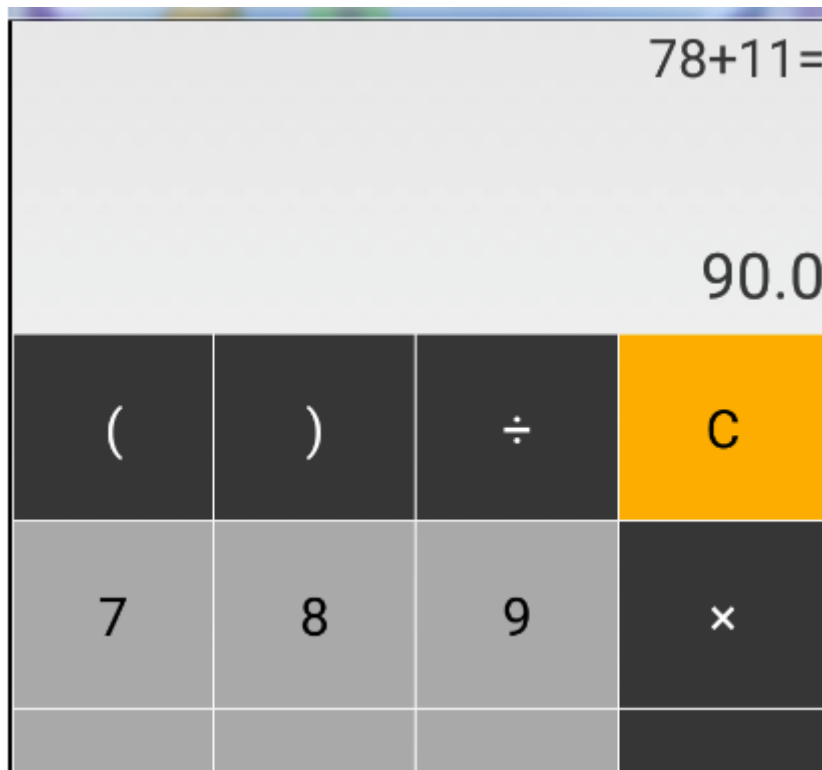
3.make Project 生成so库,如果之前没有修改/app/build.gradle文件和在local.properties中指定ndk路径则会报错。报错参考静态使用JNI中的步骤5,指定ndk路径,并且指定so名称和输出的架构。



生成成功，则在/app/build/intermediates/ndk/debug/lib目录下生成相应的so文件



#### 4.在虚拟机中运行



(注：这里的结果在原结果上加1)

#### 0x04 Android Studio加载第三方库

##### 1.使用ndk生成so文件

①建立jni目录(随便在哪个地方)

②编写so库中的.c函数，这里完全使用**0x03 Android Studio动态注册的方式使用JNI**中的c文件，使用动态注册的方式

③编写Android.mk文件(单独使用ndk编译so文件的时候需要用到)



```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_LDLIBS := -L$(SYSROOT)/usr/lib -llog
LOCAL_PRELINK_MODULE := false
```



```
LOCAL_MODULE := JniTest
LOCAL_SRC_FILES := MyJniCalc.c
LOCAL_SHARED_LIBRARIES := libandroid_runtime
include $(BUILD_SHARED_LIBRARY)
```



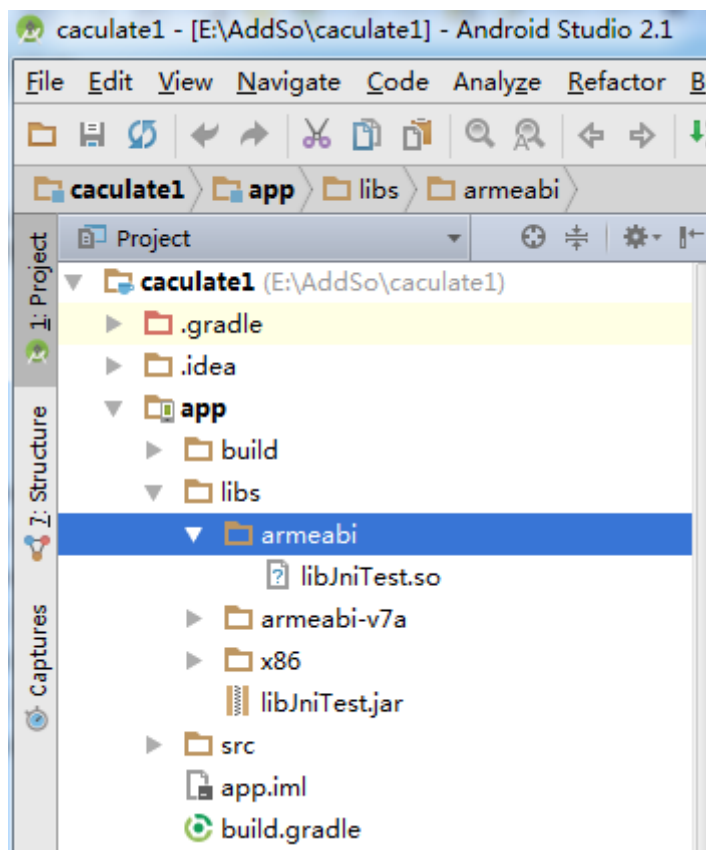
④使用cmd进入jni目录，执行ndk-build，生成so文件

```
E:\AddSo\jni>ndk-build
[armeabi] Compile thumb   : JniTest <= MyJniCalc.c
[armeabi] SharedLibrary   : libJniTest.so
[armeabi] Install         : libJniTest.so => libs/armeabi/libJniTest.so
```

2.加载生成的so文件，打包进apk中

①我们在/app目录下建立libs目录，将我们的so文件拷贝进去

这里可以使用ndk自己生成的so文件，也可以使用其他第三方so文件



②改写/app/build.gradle文件，编译的时候将so生成上图中的libJniTest.jar文件

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 16
    buildToolsVersion "23.0.3"

    defaultConfig {
        applicationId "com.example.caculate"
        minSdkVersion 8
        targetSdkVersion 16
    }
}
```

```
}

buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
    }
}

sourceSets.main {
    jni.srcDirs = []
    jniLibs.srcDir 'src/main/libs'
}

task nativeLibsToJar(type: Zip, description: "create a jar archive of the native libs") {
    destinationDir file("$projectDir/libs")
    baseName "libJniTest"
    extension "jar"
    from fileTree(dir: "libs", include: "**/*.so")
    into "lib"
}

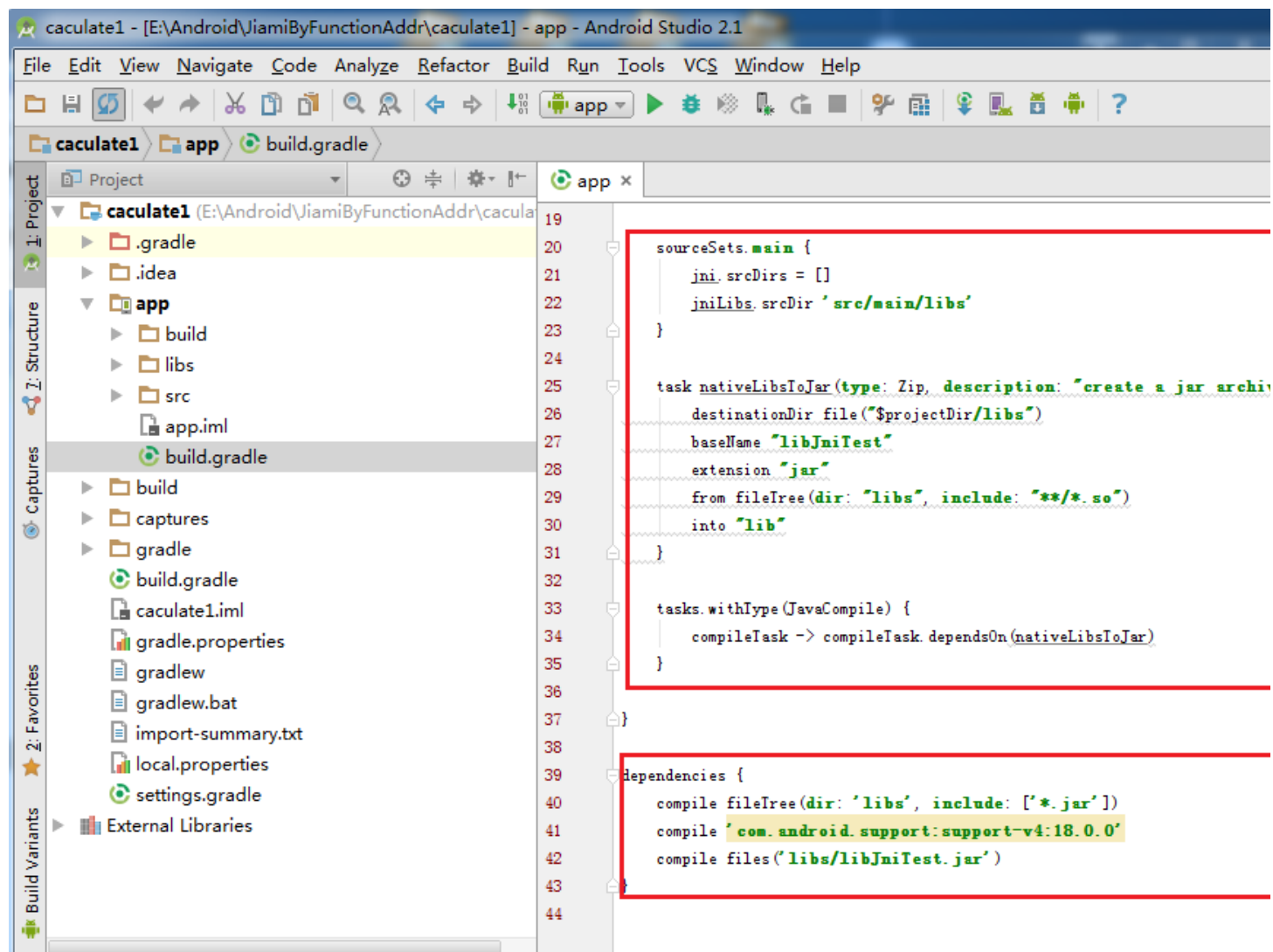
tasks.withType(JavaCompile) {
    compileTask -> compileTask.dependsOn(nativeLibsToJar)
}

}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:support-v4:18.0.0'
    compile files('libs/libJniTest.jar')
}
```



其中into "lib" 是生成的jar文件存放的目录，include: "\*\*/\*.so" 为所依赖的so文件



### ③在Java类中添加so的接口

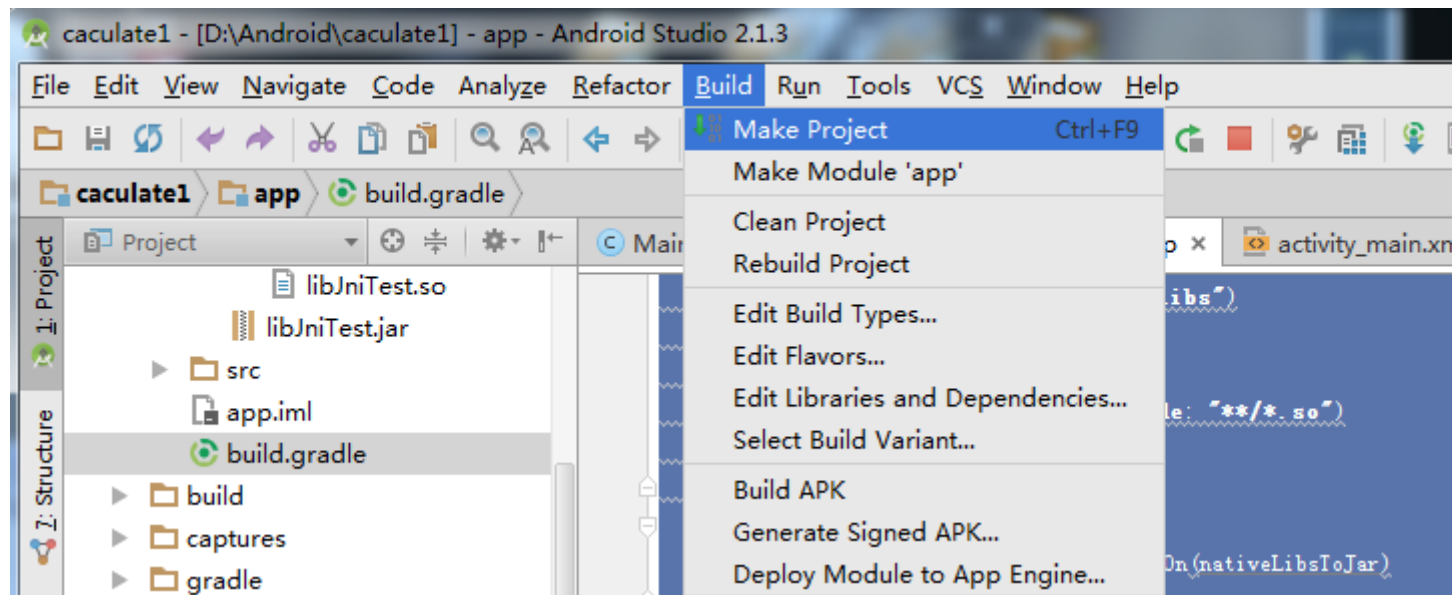


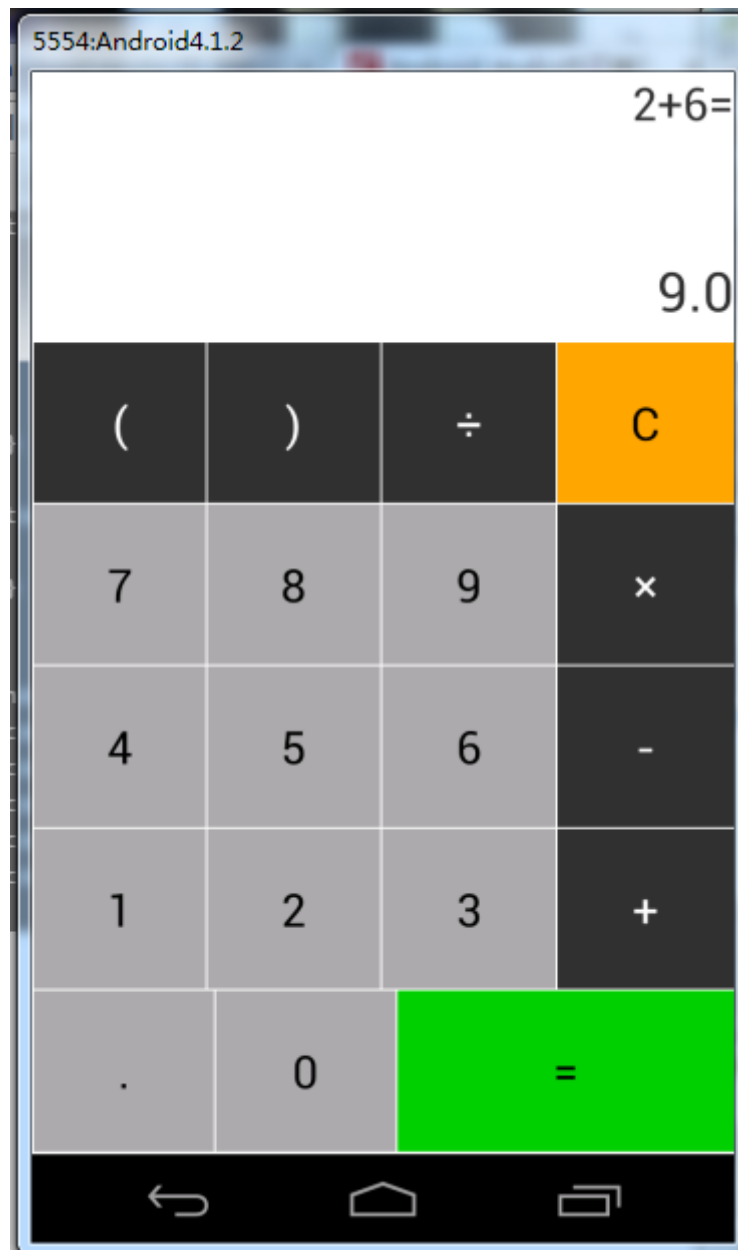
```
public class MainActivity extends Activity implements OnClickListener {  
  
    static{  
        System.loadLibrary("JniTest");  
    }  
  
    private native int Add(double num1,double num2);  
    private native int Sub(double num1,double num2);  
    private native int Mul(double num1,double num2);  
    private native int Div(double num1,double num2);  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
    }  
}
```



#### 4.Make Project , 点击运行

这里不需要指定ndk路径, so库的名称也不需要再/app/build.gradle中指定, 在Android.mk中指定





(注:这里结果在原结果上加1)

## 0x05 总结

这里是Android Studio中的使用方法，刚开始也是各种百度，弄得很复杂，在后面的学习中也懂得了很多，所以今天将使用方法重新整理了一遍，希望大家可以不要浪费太多时间。**静态注册方法**主要要用javah生成.h文件，显得比较复杂，每次添加函数都要重新生成.h文件，不过如果知道函数名称的格式也可以不生成.h文件，实践中，在jni目录下只有main.c文件也是可以运行成功的。**动态注册方法**直接在函数中注册比较容易动态扩展，但是需要对注册的数据类型有所了解，可以参考静态方法生成的.h文件中有对应的数据类型。**加载第三方库**感觉是最实用的，不管是so文件加密，还是使用不开源的so库，都需要加载已经生成的so文件。

注意/app/build.gradle中的sdk版本要改成自己android studio中sdk有的版本，如果没有也可根据android studio提示下载。

代码下载：<http://pan.baidu.com/s/1dFyWHhf>

最后编辑于2016.9.17

分类: [安卓](#)



[Alif](#)

[关注 - 7](#)

[粉丝 - 12](#)

[+加关注](#)

0

0

« 上一篇: [Win7 x64下进程保护与文件保护\(ObRegisterCallbacks\)](#)

» 下一篇: [病毒分析要掌握的技能](#)

posted @ 2016-05-19 00:33 [Alif](#) 阅读(6280) 评论(3) [编辑](#) [收藏](#)

## 评论

#1楼 2016-08-26 16:34 | 一写成名

请问一下，网盘密码是多少啊？

[支持\(0\)](#) [反对\(0\)](#)

#2楼[楼主] 2016-08-26 17:08 | [Alif](#)



@ 一写成名

现在应该好了，没密码了

支持(0) 反对(0)

#3楼 2016-10-12 16:31 | Royanss

请问有方法可以知道别人so生成的jar里面的方法么？

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】腾讯云免费实验室，1小时搭建人工智能应用

【新闻】H3 BPM体验平台全面上线



最新IT新闻:

- 遭英美封杀 卡巴斯基关闭华盛顿分支机构
- 传苹果4亿美元收购音乐识别应用Shazam后者估值10亿美元
- iPhone X需求趋于平缓 供应商订单量开始缩减
- 代购电商CEO发公开信质疑淘宝封杀|阿里回应：黄牛该封

- [马斯克承认特斯拉开发AI芯片 将用于无人驾驶系统](#)
- » [更多新闻...](#)



**最新知识库文章:**

- [以操作系统的角度述说线程与进程](#)
- [软件测试转型之路](#)
- [门内门外看招聘](#)
- [大道至简，职场上做人做事做管理](#)
- [关于编程，你的练习是不是有效的？](#)
- » [更多知识库文章...](#)

Copyright ©2017 Alif