### Mac下AndroidStudio NDK 环境搭建以及so文件使用

撰写人:Talon

目录

Mac下AndroidStudio NDK 环境搭建以及so文件使用 1

一，NDK环境配置 3

1，ndk 基本介绍参考链接 3

2，mac ndk下载链接 3

3，在cmd命令中解压文件 3

4, 配置NDK 路径pico .bash\_profile 4

5. 检查是否配置成功 5

二，NDK项目运行 6

1，新建一个android Project 6

2，构建项目成功后，执行javah 命令 7

3，编写C文件 8

4，java类中引用so文件 9

5，gradle文件中配置 10

三，移植so文件到其他项目 11

一，复制so文件到新建的项目中 11

二，配置一些基本信息 13

三，使用so文件中的方法 15

# 一，NDK环境配置

## 1，ndk 基本介绍参考链接

1. http://www.cnblogs.com/devinzhang/archive/2012/02/29/2373729.html（NDK基本介绍）
2. http://blog.csdn.net/u010350809/article/details/46840893（NDK环境配置）
3. http://blog.csdn.net/yanbober/article/details/45309049（NDK搭建项目）

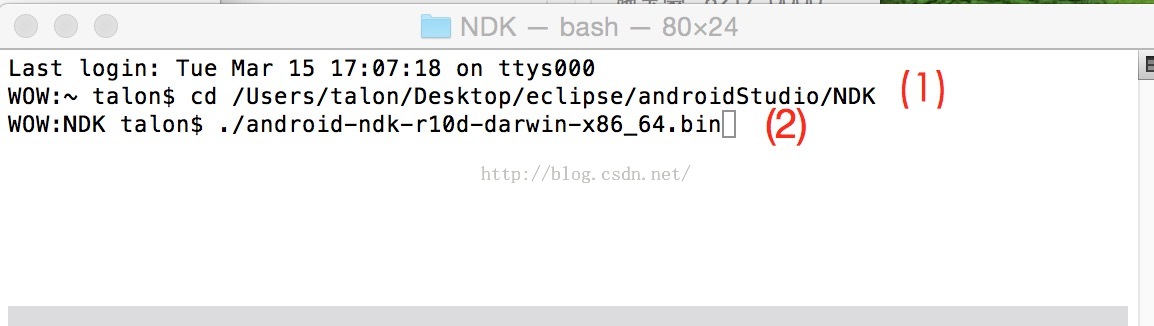
## 2，mac ndk下载链接

 https://dl.google.com/android/ndk/android-ndk-r10d-darwin-x86\_64.bin

## 3，在cmd命令中解压文件

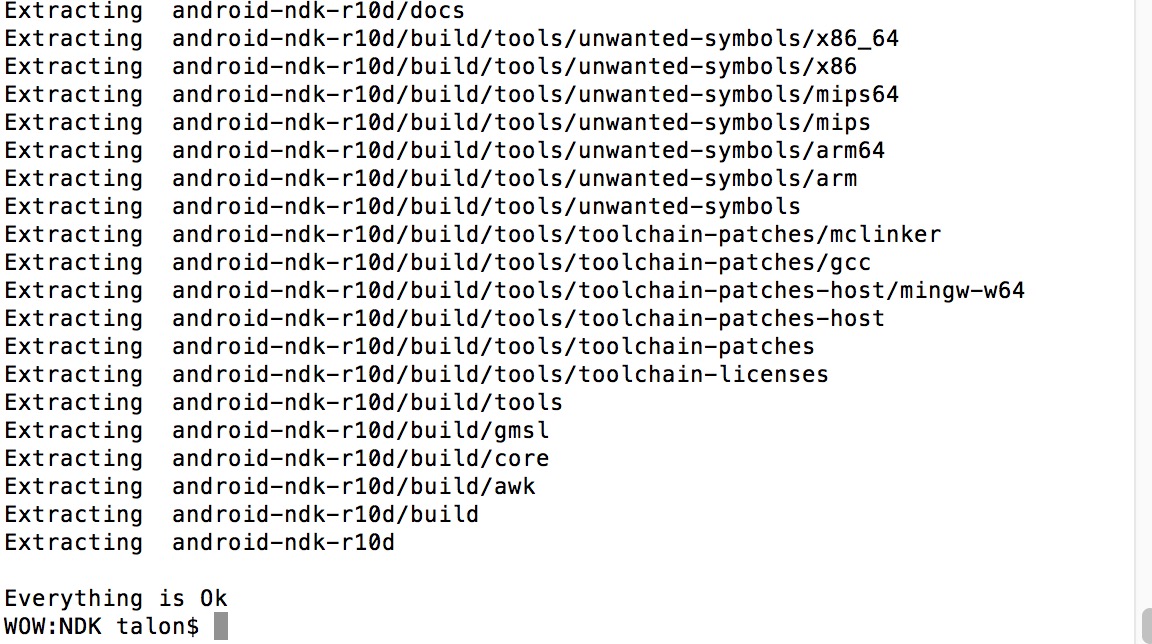
  (1) cd  命令进入下载后ndk存放的目录  cd /Users/talon/Desktop/eclipse/androidStudio/NDK

  (2)执行解压命令 ./android-ndk-r10d-darwin-x86\_64.bin



  按下回车后，可以看到一直有信息在跑。

  看到下图 代表解压成功。



 解压之后会看到一个文件夹。

## 4, 配置NDK 路径pico .bash\_profile

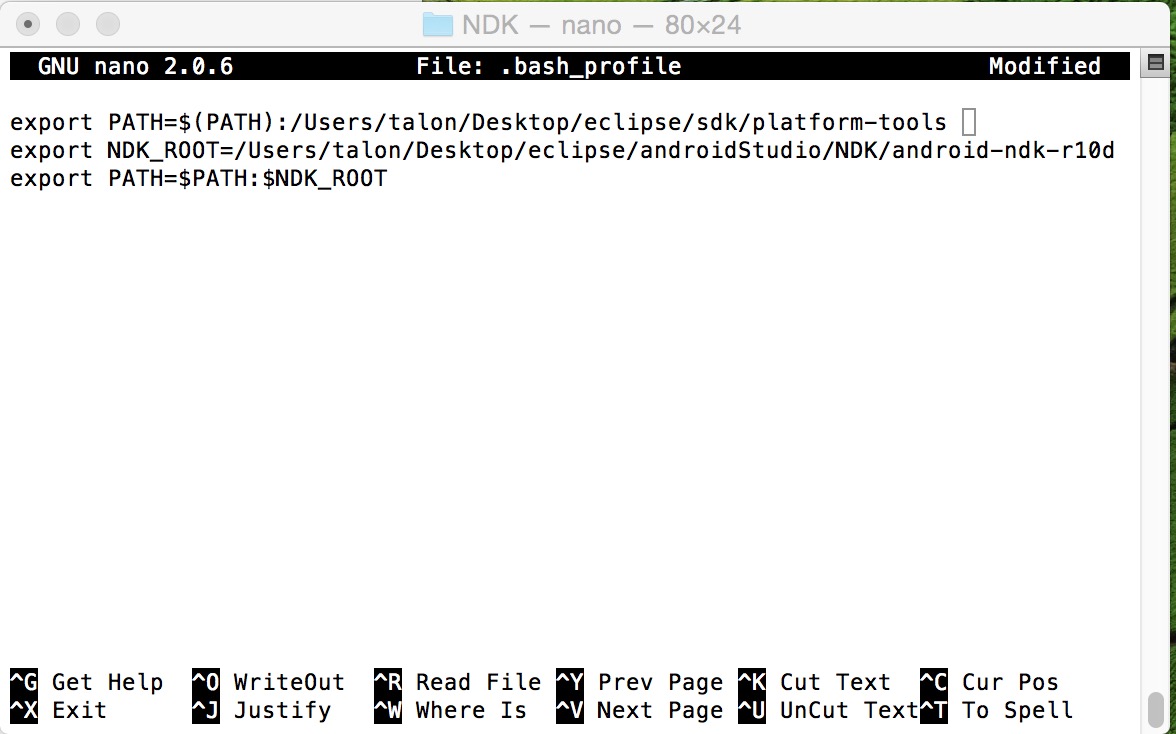
  (1), 输入  pico .bash\_profile   回车。

  (2). export PATH=$(PATH):/Users/talon/Desktop/eclipse/sdk/platform-tools

  (3). export NDK\_ROOT=/Users/talon/Desktop/eclipse/androidStudio/NDK/android-ndk-r10d

  (4). export PATH=$PATH:$NDK\_ROOT  
  最后保存（ control＋X） 选 Y

  (5). 更新刚配置的环境变量输入source .bash\_profile  (如果没有更新成功，尝试重启电脑)

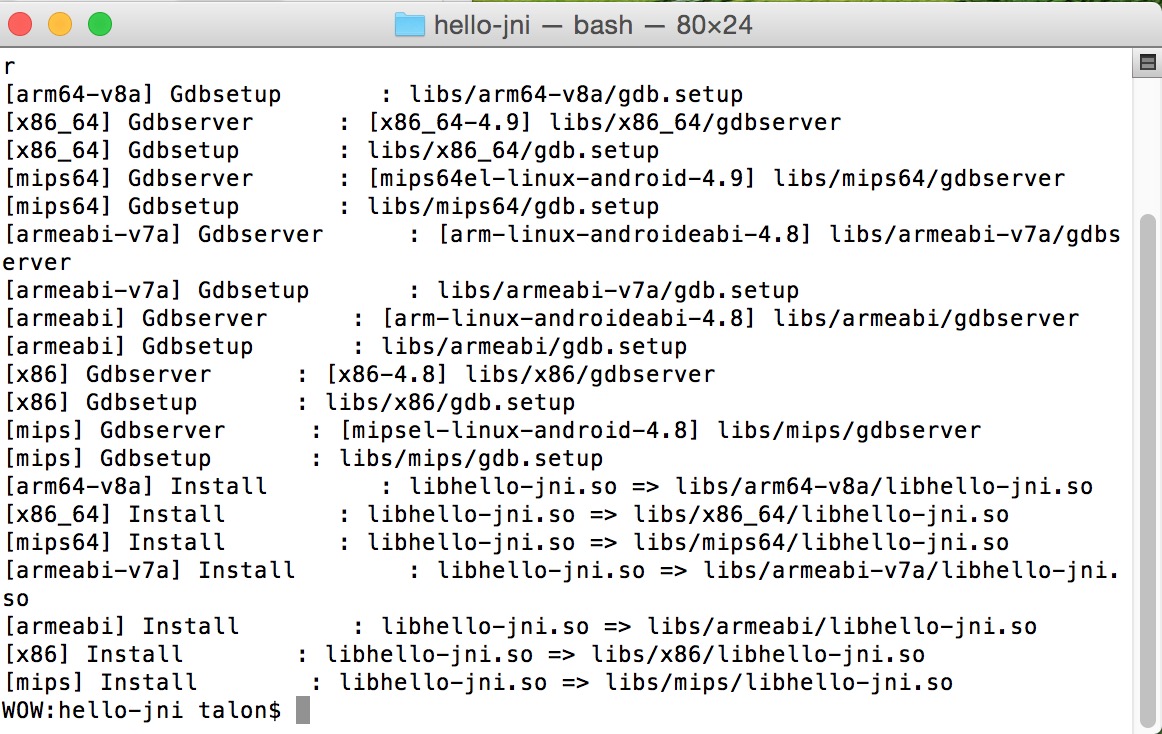


## 5. 检查是否配置成功

 (1) 终端进入到 NDK下面的 samples 目录下。

 (2) 输入 cd hello-jni/  ，回车，然后执行 ndk-build

  出现以下界面代表配置成功。



# 二，NDK项目运行

## 1，新建一个android Project

新建一个Android Project，然后新建一个java类，命名为：JniUtils

写入以下方法：

public class JniUtils {

public native String getString();

}

然后在MainActivity中调用这个方法。 将这个方法的返回值，显示在界面上。然后build project

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.*activity\_main*);

JniUtils jniUtils = new JniUtils();

TextView tv = (TextView) findViewById(R.id.*tv*); tv.setText(jniUtils.getString());

} }

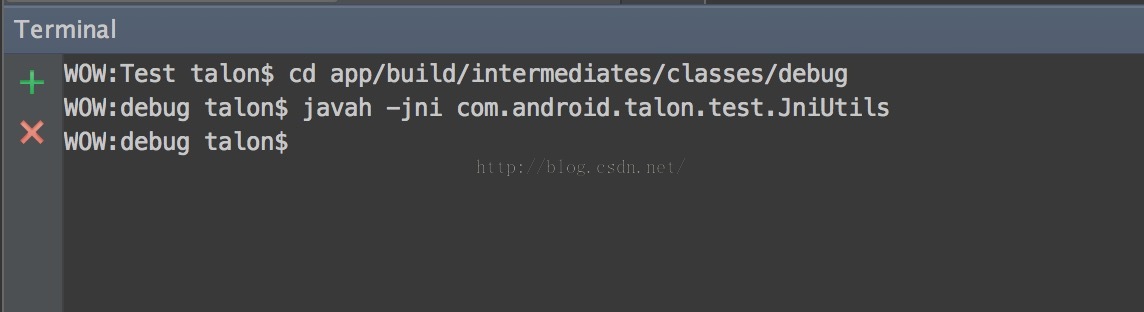
## 2，构建项目成功后，执行javah 命令

构建项目成功后。注意Test\app\build\intermediates\classes\debug 目录。

然后接下来的步骤就是根据生成的class文件，利用javah生成对应的 .h头文件。

点开AS的Terminal标签，cd命令进入到该项目的app/build/intermediates/classes/debug/ 文件夹下。

然后执行命令：javah -jni com.android.talon.test.JniUtils



然后查看文件夹 Test\app\build\intermediates\classes\debug 会生成一个.h文件：com\_android\_talon\_JniUtils.h

## 3，编写C文件

在工程的main目录下新建一个名字为jni的目录，然后将刚才的 .h文件剪切过来。在jni目录下新建一个c文件，随意取名，我的叫jnitest.c 。然后编辑代码如下

#include "com\_android\_talon\_test\_JniUtils.h"

/\*

\* Class: com\_android\_talon\_test\_JniUtils.h

\* Method: getString

\* Signature: ()Ljava/lang/String;

\*/

JNIEXPORT jstring JNICALL Java\_com\_android\_talon\_test\_JniUtils\_getString (JNIEnv \*env, jobject obj){

**return** (\*env)->NewStringUTF(env,"Hello World!");

}

接下来在工程的local.properties文件中添加NDK路径

sdk.dir=/Users/talon/Desktop/eclipse/androidStudio/sdk ndk.dir=/Users/talon/Desktop/eclipse/androidStudio/NDK/android-ndk-r10d

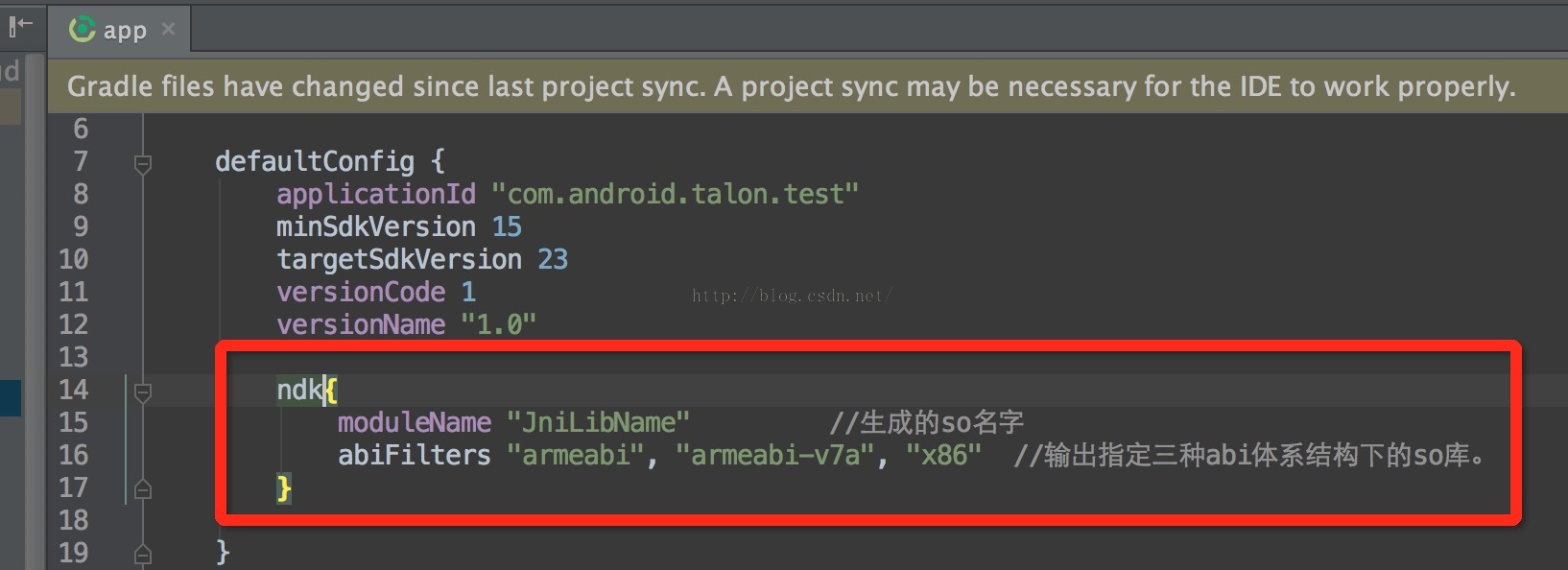
接下来在app module目录下的build.gradle中设置库文件名（生成的so文件名）。找到gradle文件的defaultConfig这项，在里面添加如下内容：

ndk{

moduleName "JniLibName" //生成的so名字 abiFilters "armeabi", "armeabi-v7a", "x86" //输出指定三种abi体系结构下的so库。

}

如图：



## 4，java类中引用so文件

生成的so文件也有了，那我们就要引用这个so文件。

在java 文件中。添加如下代码。将 java文件中的方法与so文件中的代码对应起来。

static {

System.*loadLibrary*("JniLibName"); //和生成so文件的名字对应。

}

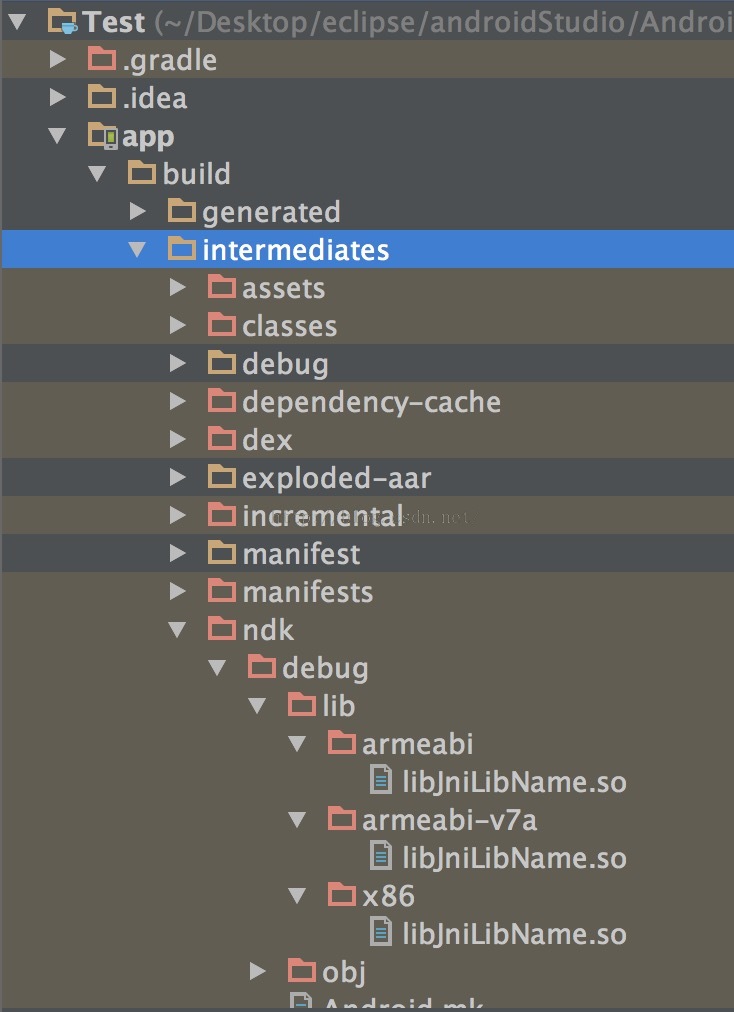
## 5，gradle文件中配置

最后在项目的 gradle.properties 文件的末尾添加如下代码：

android.useDeprecatedNdk=true

重新编译项目并运行。

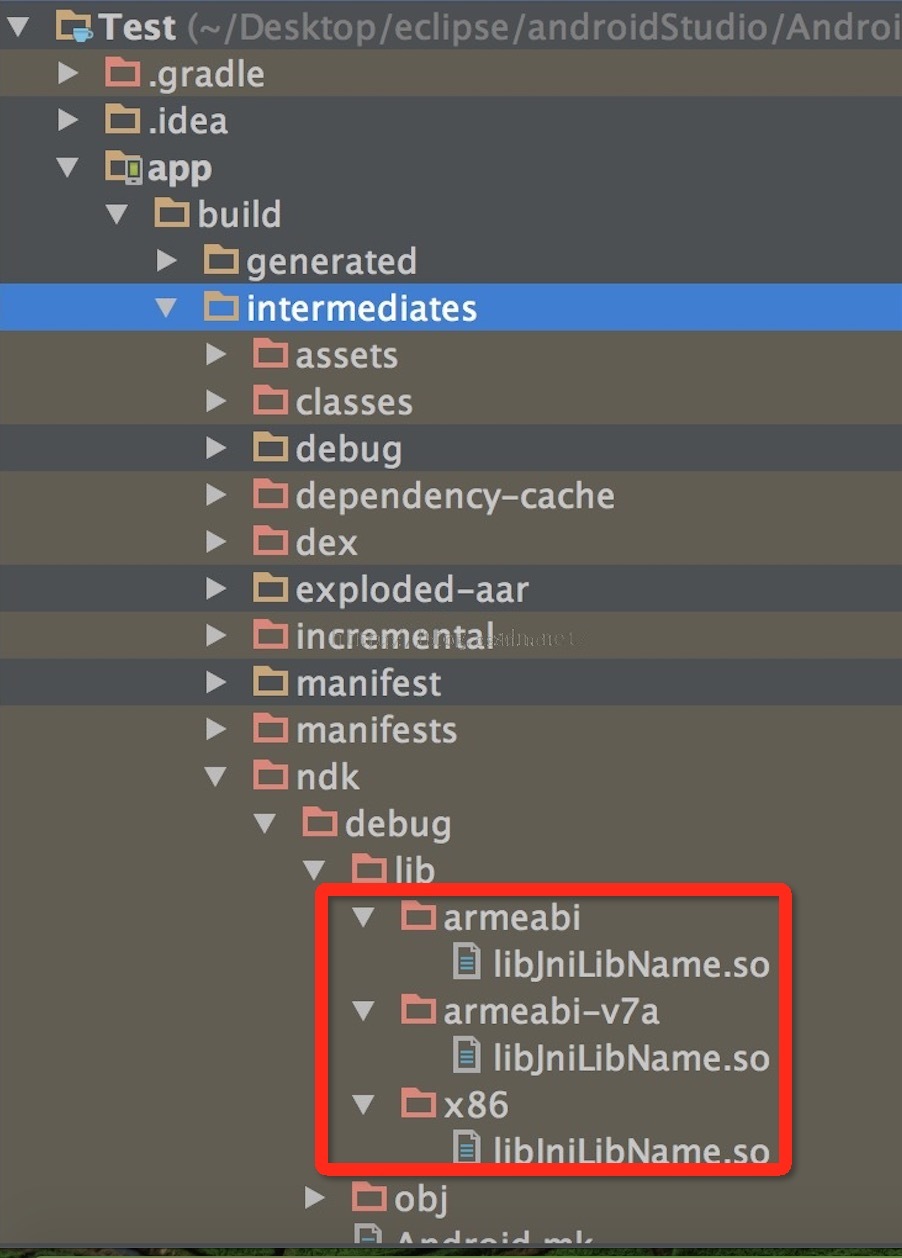
so文件生成在哪里。 请看以下截图



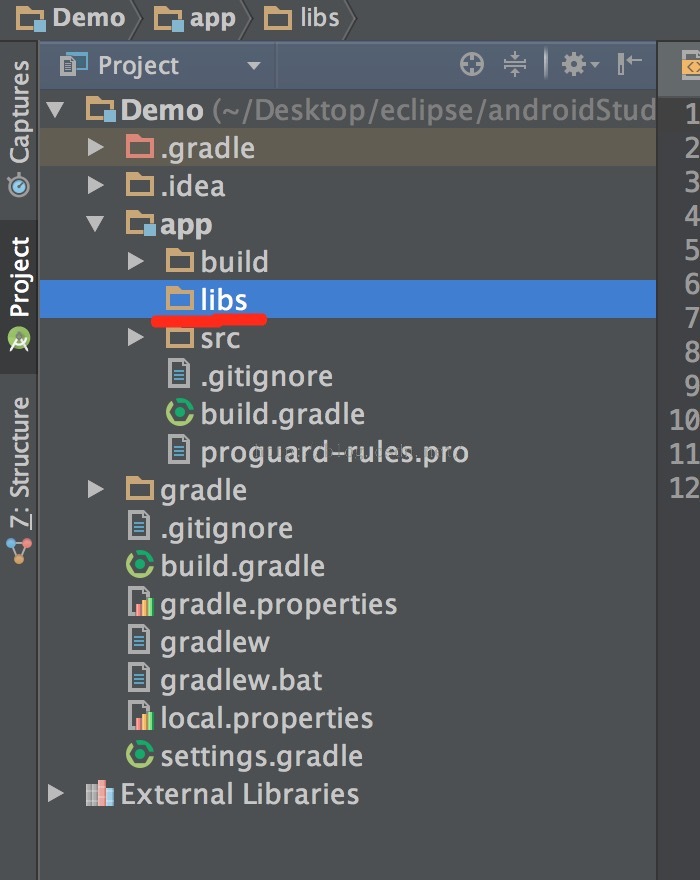
# 三，移植so文件到其他项目

## 一，复制so文件到新建的项目中

1，复制lib路径下的文件夹以及so文件。



2，新建一个项目叫Demo，将复制的文件夹和so文件复制到lib 目录下



## 二，配置一些基本信息

1，在app build.gradle 的文件的 android 结点 中加入以下代码：

sourceSets {

main {

jniLibs.srcDirs = ['libs']

}

}

如下图：



## 三，使用so文件中的方法

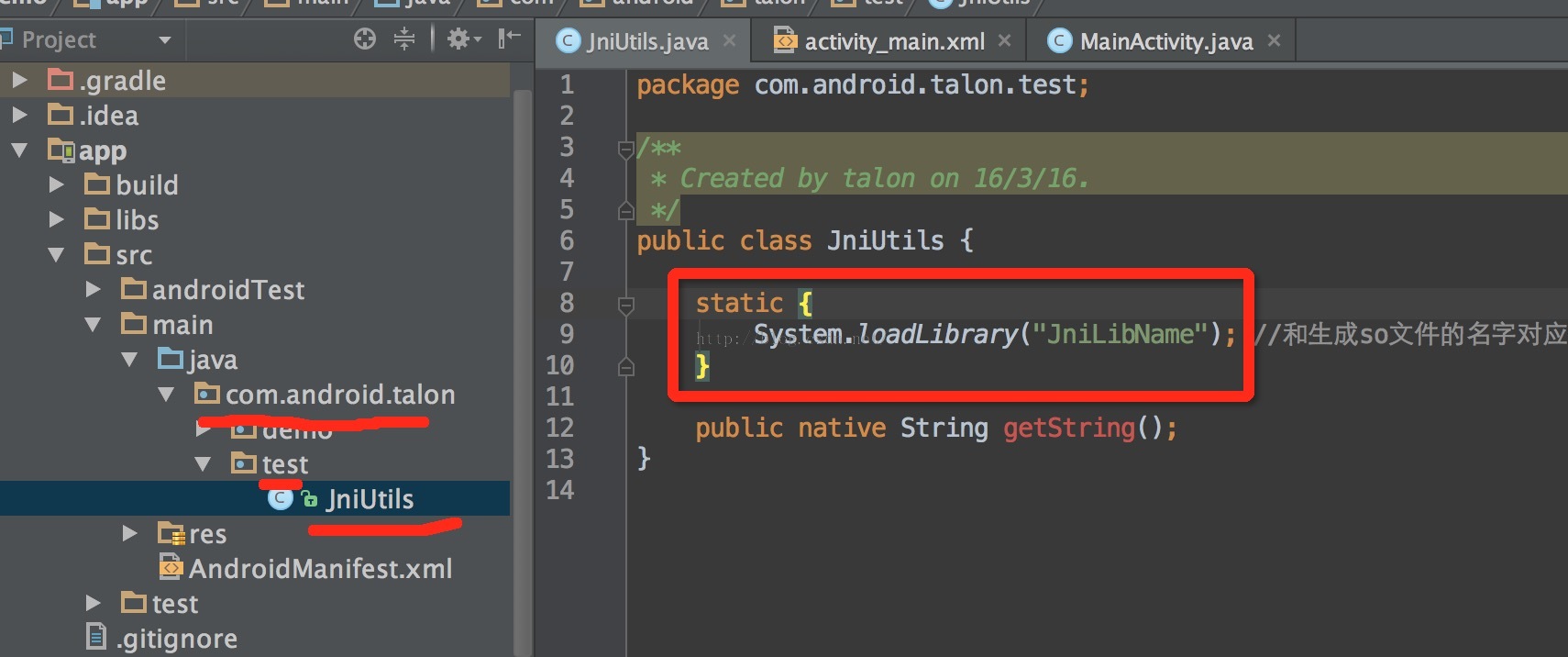
1，由于上次编译的c文件中的方法 指定了包名类名方法名。所以需要和so文件保持对应的包名类名和方法名。同时在java类中添加引用so文件代码

static {

System.*loadLibrary*("JniLibName"); //和生成so文件的名字对应。

}

public native String getString();//与so文件中方法对应



运行项目。调用so成功 ！

Over！

可能出现的错误：

1，包名或者类名与 so文件不对应，或者没有在java文件中没有loadLibrary("")方法。

java.lang.UnsatisfiedLinkError: Native method not found: com.android.talon.test1.JniUtils.getString:()Ljava/lang/String;

2，app/ builed.gradle 中没有添加如下代码：

sourceSets {

main {

jniLibs.srcDirs = ['libs']

}

}

java.lang.UnsatisfiedLinkError: Couldn't load JniLibName from loader dalvik.system.PathClassLoader