

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

撰写人:Talon

目录

<i>Mac 下 AndroidStudio NDK 环境搭建以及 so 文件使用</i>	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

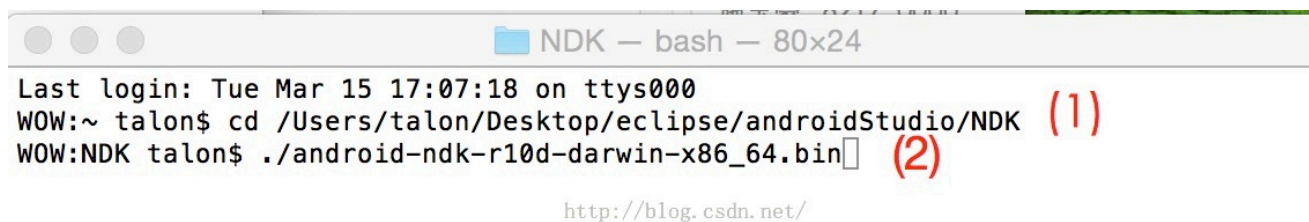
- (1) <http://www.cnblogs.com/devinzhang/archive/2012/02/29/2373729.html> (NDK 基本介绍)
- (2) <http://blog.csdn.net/u010350809/article/details/46840893> (NDK 环境配置)
- (3) <http://blog.csdn.net/yanbober/article/details/45309049> (NDK 搭建项目)

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



```
NDK — bash — 80x24
Last login: Tue Mar 15 17:07:18 on ttys000
WOW:~ talon$ cd /Users/talon/Desktop/eclipse/androidStudio/NDK (1)
WOW:NDK talon$ ./android-ndk-r10d-darwin-x86_64.bin (2)
```

<http://blog.csdn.net/>

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。

```

Extracting android-ndk-r10d/docs
Extracting android-ndk-r10d/build/tools/unwanted-symbols/x86_64
Extracting android-ndk-r10d/build/tools/unwanted-symbols/x86
Extracting android-ndk-r10d/build/tools/unwanted-symbols/mips64
Extracting android-ndk-r10d/build/tools/unwanted-symbols/mips
Extracting android-ndk-r10d/build/tools/unwanted-symbols/arm64
Extracting android-ndk-r10d/build/tools/unwanted-symbols/arm
Extracting android-ndk-r10d/build/tools/unwanted-symbols
Extracting android-ndk-r10d/build/tools/toolchain-patches/mclinker
Extracting android-ndk-r10d/build/tools/toolchain-patches/gcc
Extracting android-ndk-r10d/build/tools/toolchain-patches-host/mingw-w64
Extracting android-ndk-r10d/build/tools/toolchain-patches-host
Extracting android-ndk-r10d/build/tools/toolchain-patches
Extracting android-ndk-r10d/build/tools/toolchain-licenses
Extracting android-ndk-r10d/build/tools
Extracting android-ndk-r10d/build/gmsl
Extracting android-ndk-r10d/build/core
Extracting android-ndk-r10d/build/awk
Extracting android-ndk-r10d/build
Extracting android-ndk-r10d

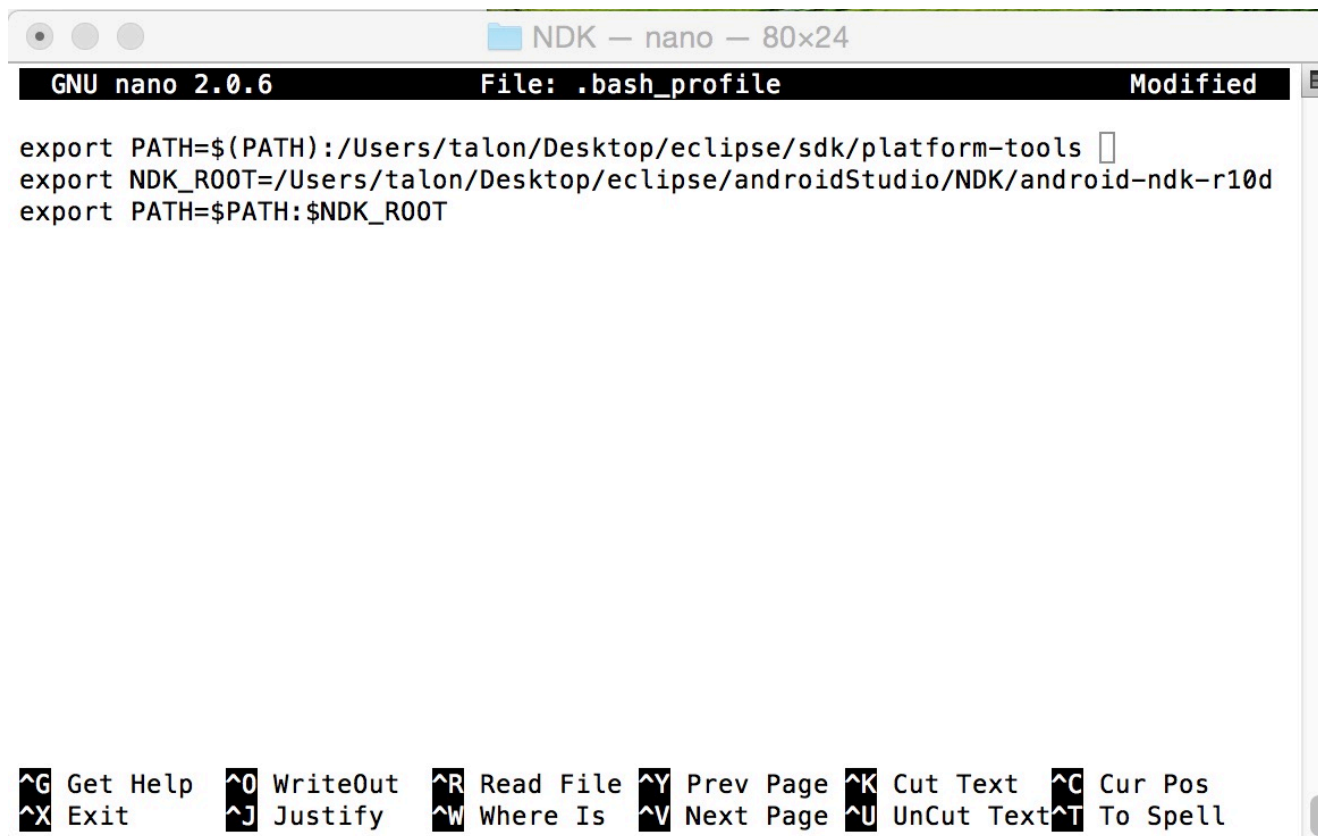
Everything is Ok
WOW:NDK talon$ █

```

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)



```
GNU nano 2.0.6 File: .bash_profile Modified
export PATH=$(PATH):/Users/talon/Desktop/eclipse/sdk/platform-tools
export NDK_ROOT=/Users/talon/Desktop/eclipse/androidStudio/NDK/android-ndk-r10d
export PATH=$PATH:$NDK_ROOT

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

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。

```

r
[arm64-v8a] Gdbsetup      : libs/arm64-v8a/gdb.setup
[x86_64] Gdbserver       : [x86_64-4.9] libs/x86_64/gdbserver
[x86_64] Gdbsetup        : libs/x86_64/gdb.setup
[mips64] Gdbserver       : [mips64el-linux-android-4.9] libs/mips64/gdbserver
[mips64] Gdbsetup        : libs/mips64/gdb.setup
[armeabi-v7a] Gdbserver  : [arm-linux-androideabi-4.8] libs/armeabi-v7a/gdbserver
[armeabi-v7a] Gdbsetup   : libs/armeabi-v7a/gdb.setup
[armeabi] Gdbserver      : [arm-linux-androideabi-4.8] libs/armeabi/gdbserver
[armeabi] Gdbsetup       : libs/armeabi/gdb.setup
[x86] Gdbserver          : [x86-4.8] libs/x86/gdbserver
[x86] Gdbsetup           : libs/x86/gdb.setup
[mips] Gdbserver         : [mipsel-linux-android-4.8] libs/mips/gdbserver
[mips] Gdbsetup          : libs/mips/gdb.setup
[arm64-v8a] Install     : libhello-jni.so => libs/arm64-v8a/libhello-jni.so
[x86_64] Install        : libhello-jni.so => libs/x86_64/libhello-jni.so
[mips64] Install        : libhello-jni.so => libs/mips64/libhello-jni.so
[armeabi-v7a] Install   : libhello-jni.so => libs/armeabi-v7a/libhello-jni.so
[armeabi] Install       : libhello-jni.so => libs/armeabi/libhello-jni.so
[x86] Install           : libhello-jni.so => libs/x86/libhello-jni.so
[mips] Install          : libhello-jni.so => libs/mips/libhello-jni.so
WOW:hello-jni talon$

```

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

Terminal

```
+ WOW:Test talon$ cd app/build/intermediates/classes/debug  
WOW:debug talon$ javah -jni com.android.talon.test.JniUtils  
✗ WOW:debug talon$
```

<http://blog.csdn.net/>

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

在工程的主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 jnittest.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-r1
0d
```


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。

}
```

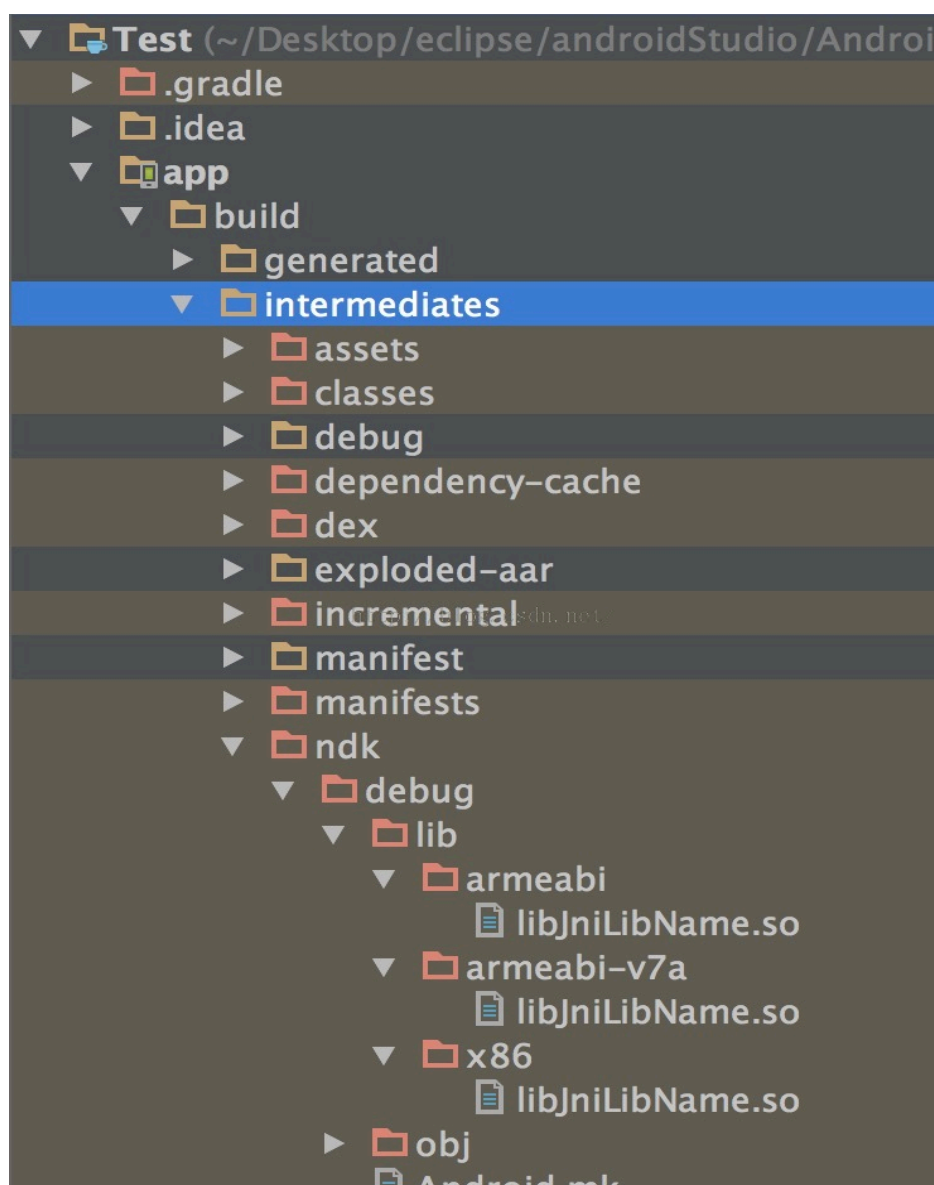
5, gradle 文件中配置

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。

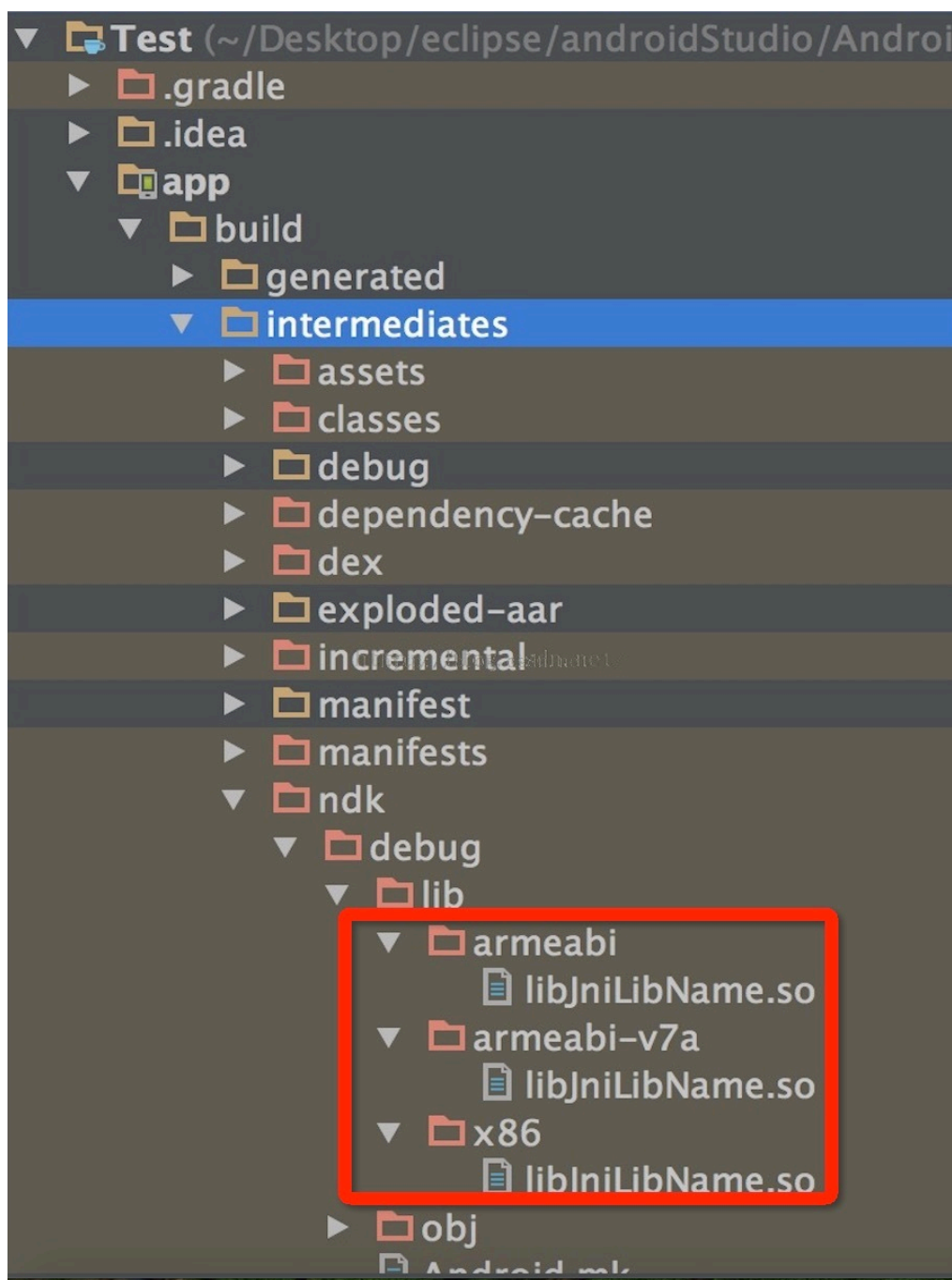
so 文件生成在哪里。 请看以下截图



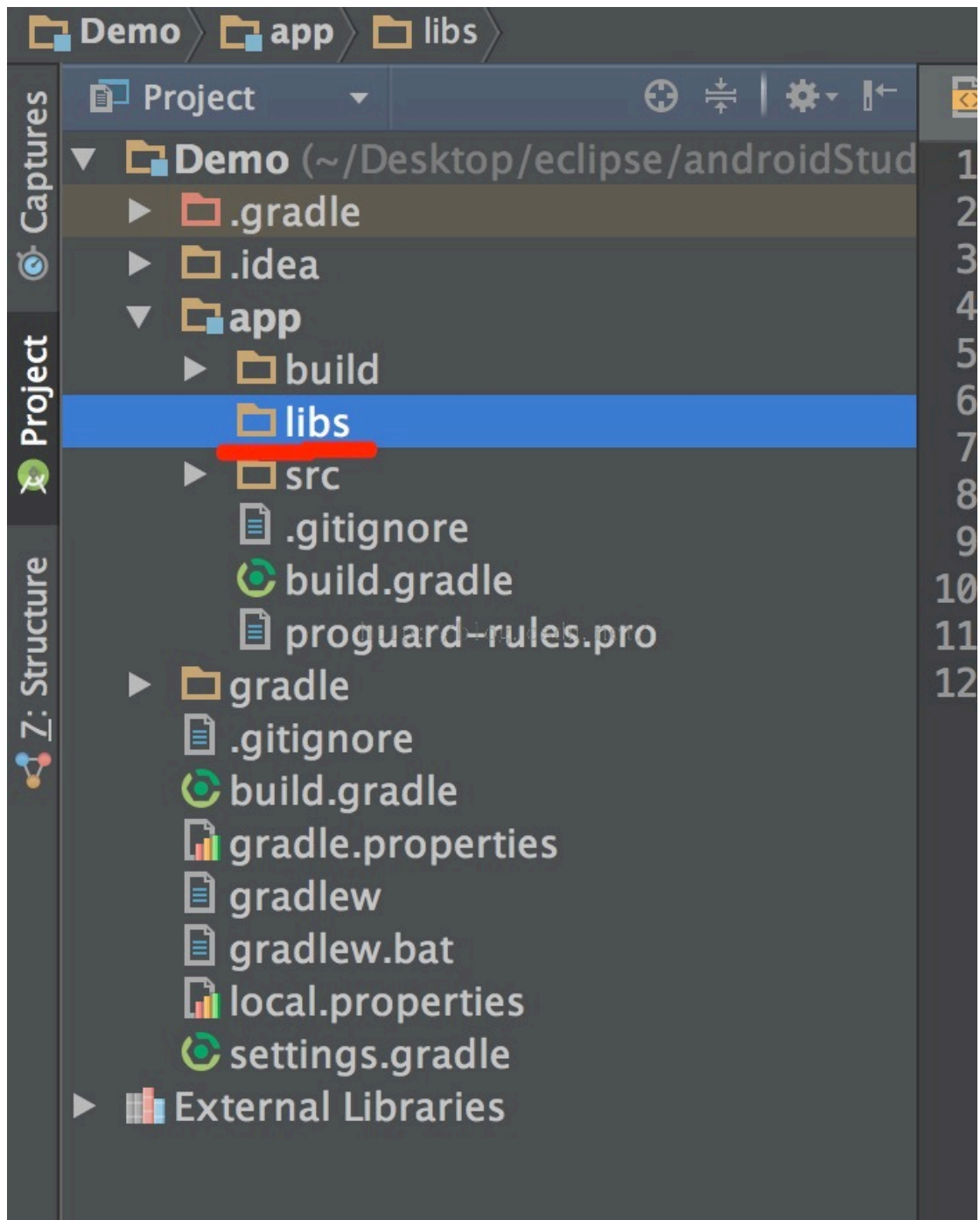
三，移植 so 文件到其他项目

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：



```

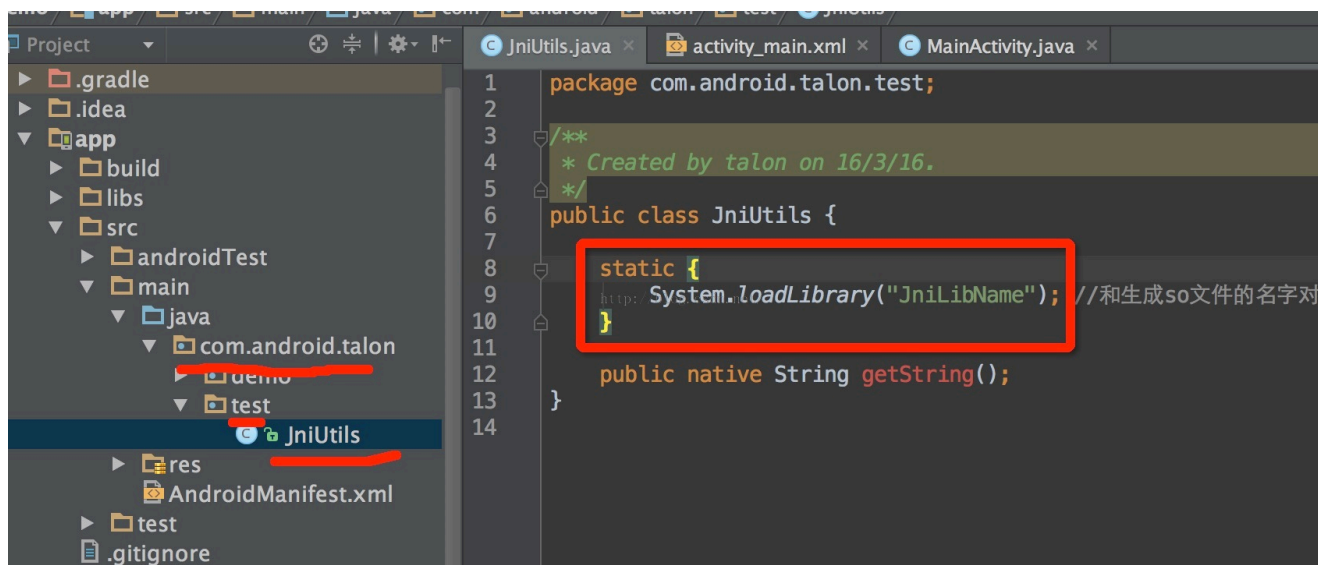
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 23
5      buildToolsVersion "23.0.2"
6
7      defaultConfig {
8          applicationId "com.android.talon.demo"
9          minSdkVersion 15
10         targetSdkVersion 23
11         versionCode 1
12         versionName "1.0"
13     }
14     buildTypes {
15         release {
16             minifyEnabled false
17             proguardFiles getDefaultProguardFile('proguard-
18                 http://blog.csdn.net/
19         }
20     }
21     sourceSets {
22         main {
23             jniLibs.srcDirs = ['libs']
24         }
25     }
26 }
27
28 dependencies {
29     compile fileTree(dir: 'libs', include: ['*.jar'])
30     testCompile 'junit:junit:4.12'
31     compile 'com.android.support:appcompat-v7:23.0.1'
32 }
33
  
```

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`