



论坛首页

版主墙

最有价值午饭

最新热帖

推荐帖子

资料下载

NEW

求职招聘

视频学院

经典版首页

[论坛首页](#) > [移动平台](#) > [Android开发论坛](#) > [史上最详细Android Studio + NDK范例](#)

[我的帖子](#) [个人中心](#) [设置](#)

◀ [返回列表](#)

➔ [高级回复](#)

+ [新帖](#) ▾

查看: **2353** | 回复: **12**

史上最详细Android Studio + NDK范例 [\[复制链接\]](#)

grayhat ▾

发表于 2016-1-14 22:30 | 来自 [51CTO网页](#)

[\[只看他\]](#) 楼主

新新人类 ☆

帖子 [13](#)

精华 [0](#)

无忧币 [52](#)

个人空间 发短消息

家园好友 他的博客




他的资源

他的课程中心

【本范例所采用的配置】

- 系统：Windows7 旗舰版，Service Pack 1，32位（最新的NDK已不支持WindowsXP）
- JDK（java包）：1.7版
- Android Studio（制作安卓程序的主要工具）：1.4版
- SDK（安卓开发工具包）：Android Studio 1.4自带的
- NDK（原生开发工具包，用来做安卓程序的C/C++部分）：用Android Studio 1.4内置的链接下载
- Experimental Plugin（一个实验版插件，目前NDK必不可少的助手）：NDK自带的
- gradle（负责安卓程序的编译）：2.5版（目前NDK只支持gradle2.5，版本高了低了都不行）

上述工具，除了Windows7，共有五个，但有些工具是捆绑在别的工具上的，所以，如果你的机器上一个也没有，要下载的只是这三个：

 android-studio-bundle-141.2288178-windows.exe
 gradle-2.5-all.zip
 jdk-7u71-windows-i586.exe

JDK在网上很容易搜到。另外两个，你可以到<https://developers.google.com/>下载，如果google的网站上去，国内有一个网站<http://www.androiddevtools.cn/>收录了绝大部分安卓开发工具。

分享至:



0

收藏 



楼主关注

二进制在虚拟社交平台中的应用——科技创新与应用

cocos2d android 1从哪里下载

Monthview,ViewSwitch请教大神

【教程】搭配AS，如何实现App远程真机debug（附：老
安卓手机酷狗控件图片

你用什么android开发工具和什么开发语言

版主推荐

Android ListView实现不同item的方法和原理分析

Android LocalBroadcastManager使用方法和代码流程分

Android第二期 - 动画数字三元归一

安卓智能聊天机器人的实现及源码分享

求助android sdk manager中无法下载新的sdk api

我是互联网公司运营的，公司APP的一个体验问题。

grayhat

发表于 2016-1-14 22:37 | 来自 51CTO网页

[只看他] 沙发



新新人类

帖子 13

精华 0

无忧币 52

个人空间 发短消息

家园好友 他的博客

他的资源

他的课程中心

【安装】

假如你的系统从来没有碰过Android，要做的事情是：

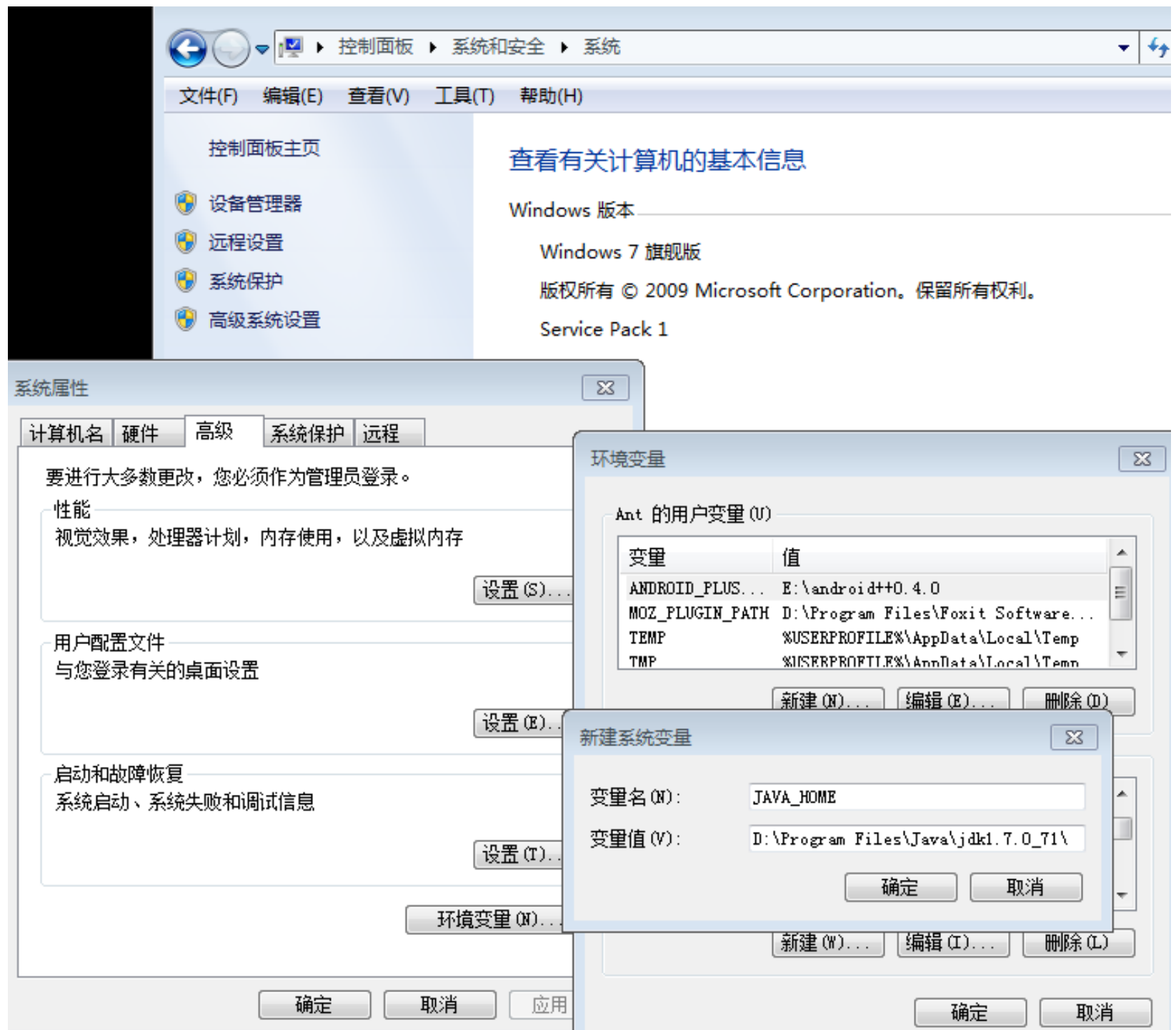
一、安装JDK 1.7

重要的是记住安装路径。我的电脑是双系统，Windows7在D盘上，所以我装java的路径是“D:\Program Files\Java\jdk1.7.0_71”。

过去，在WindowsXP中使用Android Studio，装java要避免带空格的路径，现在Windows7没有这个限制了，你按默认的路径安装即可。

二、给java设环境变量

在电脑桌面左下角点“开始”按钮，然后依次选“控制面板”、“系统和安全”、“系统”、“环境变量”，打开“环境变量”对话框，这里有两个“新建”按钮，点下面那个（再次强调，是下面那个），建一个新的系统变量，名为“JAVA_HOME”，值为java的安装路径（我的是“D:\Program Files\Java\jdk1.7.0_71”）。



再建一个新的系统变量，名为“CLASSPATH”，值为“.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar”，注意前面有点和分号。

找到已有的系统变量“PATH”，双击它，打开编辑它的窗口，在变量值的末尾加“;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin”，注意前面有分号。

总结刚才的3个环境变量：

·JAVA_HOME（新建的） java的安装路径

·CLASSPATH（新建的）

.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar

·PATH（改原来的） ;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin（加在原值的后面）

改完之后一连串的“确定”，使这些变量得以保存。

验证java是否装好的方法：在DOS窗口中输入java -version回车，若看到版本信息，就是装好了。

三、安装AndroidStudio1.4及NDK

安装过程略。从提示文字上可以看到，这个版本Android Studio把SDK也一并安装到你电脑里了。

若不带SDK，就要单独下载和安装SDK，过后还要在Android Studio中填写SDK安装路径。

装好之后先别着急启动，在Android Studio的安装目录中找到bin文件夹，在其中找到idea.properties，用记事本打开，在其末尾添加一行并保存：

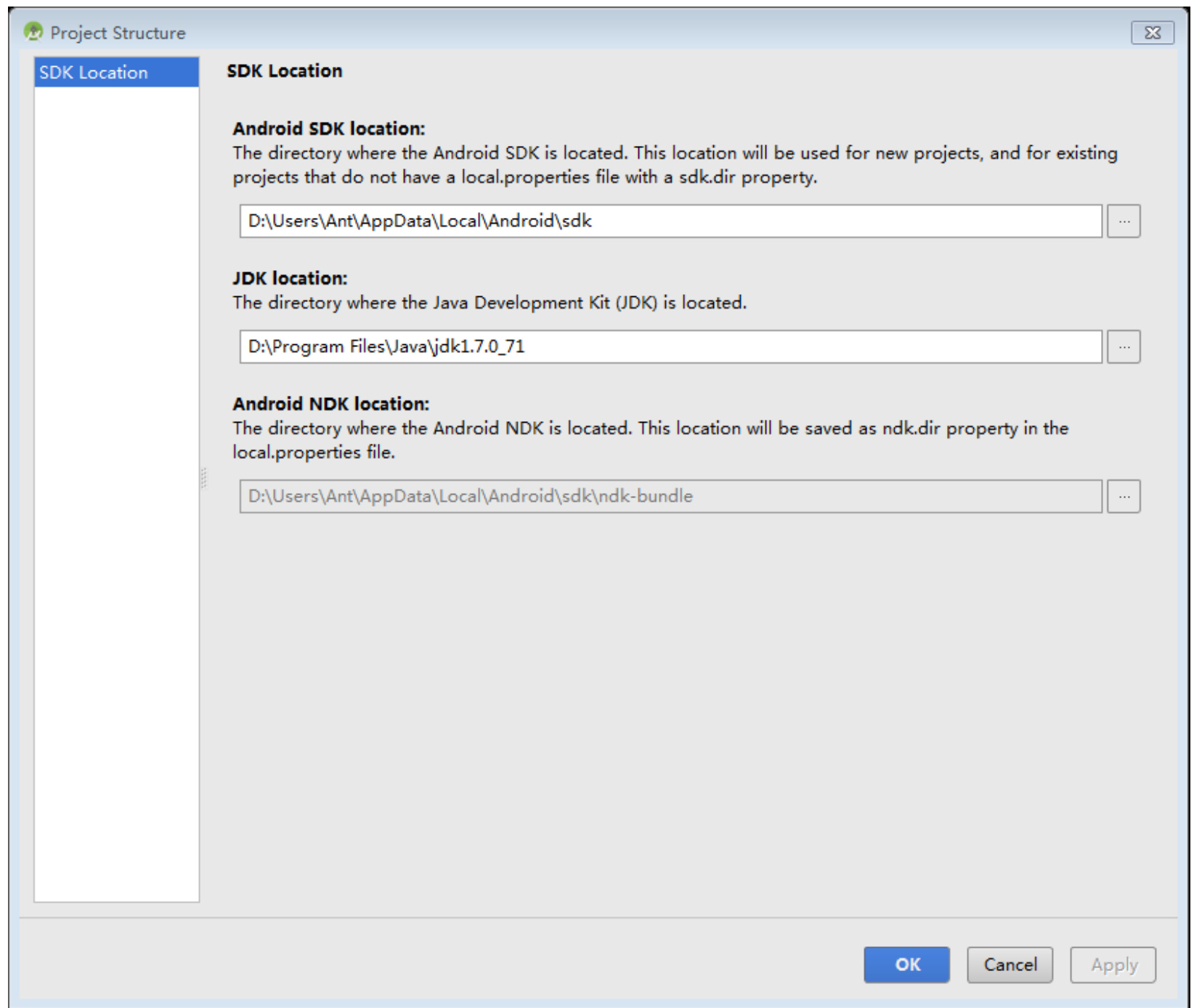
```
disable.android.first.run=true
```

如果Windows7不让你修改这个位于系统盘的文件，那就把它拷贝到别的地方修改，再拷回去覆盖原文件，这是可以的，因为Windows7允许系统盘更换文件，只不过先问问你是不是管理员。

做这件事，是为了防止Android Studio启动时不停地连google服务器（在不翻墙的情况下根本连不上，只能让程序停在那儿不动）。

然后启动Android Studio，如果走不动，多半是因为java没装好。

出现“Welcome...”窗口后，选“Configure”、“Project Defaults”、“Project Structure”，打开“Project Structure”窗口：



你第一次启动时看到的窗口不是这样的，“Android NDK location”中没有东西，可能“JDK location”中也没有。

“Android SDK location”肯定有了，因为SDK是这个版本自带的，它装好了，路径也就自动填上了。但JDK可能需要你手工填写，把java的安装路径填进去（也就是刚才设环境变量“JAVA_HOME”时填的路径）。

至于NDK，先要安装。

尚未安装NDK时，在此窗口的“Android NDK location”下会有一个按钮让你安装，点它按提示进行，在翻墙的情况下，经过漫长的等待，Android Studio告诉你在下SDK，其实也在下NDK，下载完在提示文字中就看到了，这是NDK。接着进入NDK安装，这用不了多久。装好后就自动填上了NDK的路径，就成了上图的样子。

有人说翻墙麻烦，不如找一个国内的链接下载NDK，安装，把地址告诉Android Studio。但这样一来，只能在项目中填写NDK地址，不能在整个程序中固定它。

而且google的官网建议用Android Studio内置的链接下载NDK，版本是r10e，必须装在SDK目录下的ndk-bundle文件夹中，配套的gradle只能是2.5版，SDK至少是19.0.0版且带生成工具（参阅tools.android.com/tech-docs/new-build-system/gradle-experimental），既然这么麻烦，还不如直接用Android Studio内置的链接下载。

如果你永远不需要在项目中写C或C++代码，就不用管NDK了，gradle也就用Android Studio自带的就行了，下一步也就免了。

在这里还要为NDK设环境变量：

NDK_ROOT（新建） NDK的安装路径

PATH（结尾增加） ;%NDK_ROOT%

AD 51CTO社区：活动汇总【9个活动正在进行中】 | #论坛扒点档#，随你造！

grayhat ▼

发表于 2016-1-14 22:40 | 来自 [51CTO网页](#)

[只看他] 板凳

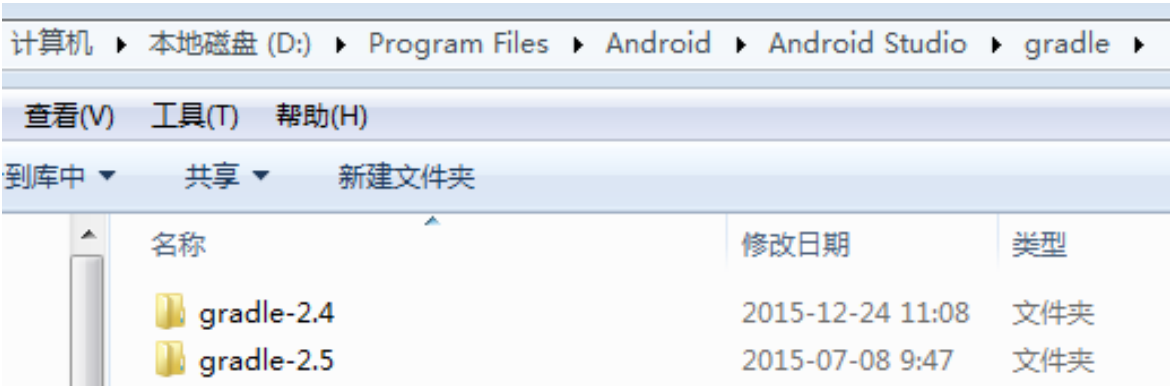


新新人类 🌟

帖子 [13](#)

四、安装gradle

下载的gradle是个压缩文件，把它解压成一个文件夹，放到Android Studio自带的gradle文件夹旁边，像这样：



此图中，gradle-2.4是Android Studio 1.4原配的，gradle-2.5即将取代它的，是NDK要求的。

五、安装手机驱动

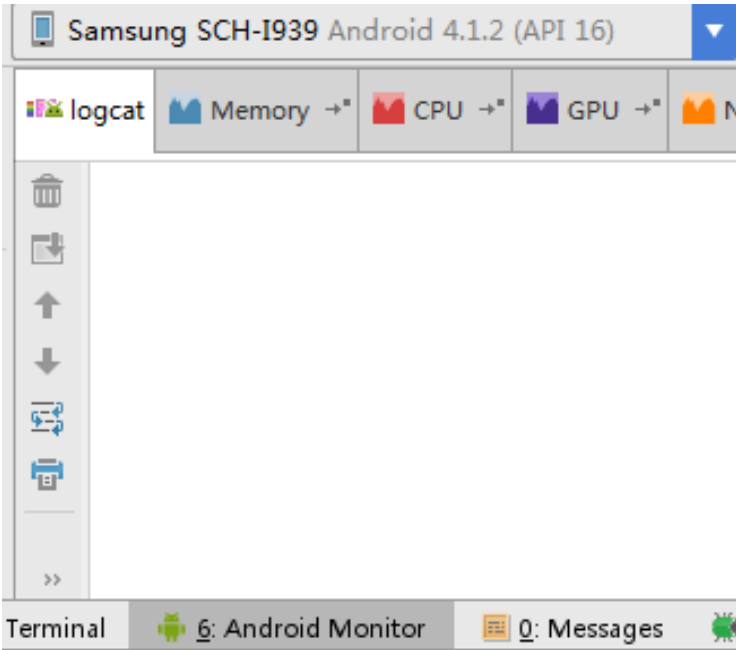
Android Studio有模拟器供你调试，但最好用真机，一是真机调试快，二是能表现所有功能、暴露所有问题。

1. 把安卓手机用数据线连在电脑的USB口上，就是你充电用的那根线，把插头拔下来，只用线，线的细的那头插在手机上，粗的那头插在电脑的USB口上。
2. 在手机的“开发者选项”中勾选“USB调试”。不同的手机品牌或安卓版本，这个选项的位置有所不同，我手头这个手机，点“设定”按钮后可以看到一串选项的底部有“开发者选项”，点开它可以看到“USB调试”这个选项。你的手机怎么样，自己找找吧。
3. 安装该手机的驱动程序。可以装“360手机助手”、“91手机助手”之类的，它发现电脑连上手机，就会自动下载该手机的驱动程序，当它显示手机型号时，驱动就装好了，360手机助手是把手机型号显示在左上角的。这时手机上也会出现USB图标，拉开它会看见“已连接为媒体设备”。

还有一个迹象表明手机驱动装好了——电脑的设备管理器显示“Android Phone”

Android Studio第一次识别手机可能比较慢，可能要依赖“360手机助手”这样的软件来装手机驱动，可能在驱动装好之后还是找不到手机，但重启电脑就找到了。或者还有种种稀奇古怪的问题，来回折腾碰巧哪一次找到了，以后就能找到了。手机软件开发就是这样，搞不明白就折腾，碰运气，奇怪的是问题总能解决，要真是永远解决不了倒好了，再

也不用受它的气了。
说了这么多，就是一连手机，二装驱动，三看软件找到手机没有。
找到手机，Android Studio会显示手机型号，在界面下方点“Android Monitor”选项卡，就可以看到手机型号。



AD 51CTO社区：活动汇总【9个活动正在进行中】 | #论坛扒点档#，随你造！

发表于 2016-1-14 22:50 | 来自 51CTO网页 [只看他] 地板

【入门练习】

一、建新项目

grayhat



新新人类




帖子 [13](#)


精华 [0](#)

无忧币 [52](#)

个人空间  发短消息

 家园好友  他的博客

 他的资源

 他的课程中心

重新启动Android Studio，在“Welcome...”窗口中选“Start a new Android Studio project”，按提示一步一步“Next”。首先是起名：

Application name（应用名）：就是你要做的程序的名称，也将是项目名称。

Company Domain（公司名）：一般的格式是“类别.部门.公司”，这个练习只使用了“类别.部门”，填的是“exercise.myself”，意思就是“我自己.练习”。

你修改上面两项，软件就会自动更新下面的一项：

Package name（包名）：这很重要，是你的程序的标识，C/C++代码会引用它。你不必现在记住它，因为写代码时随时可以在java代码中查到它。

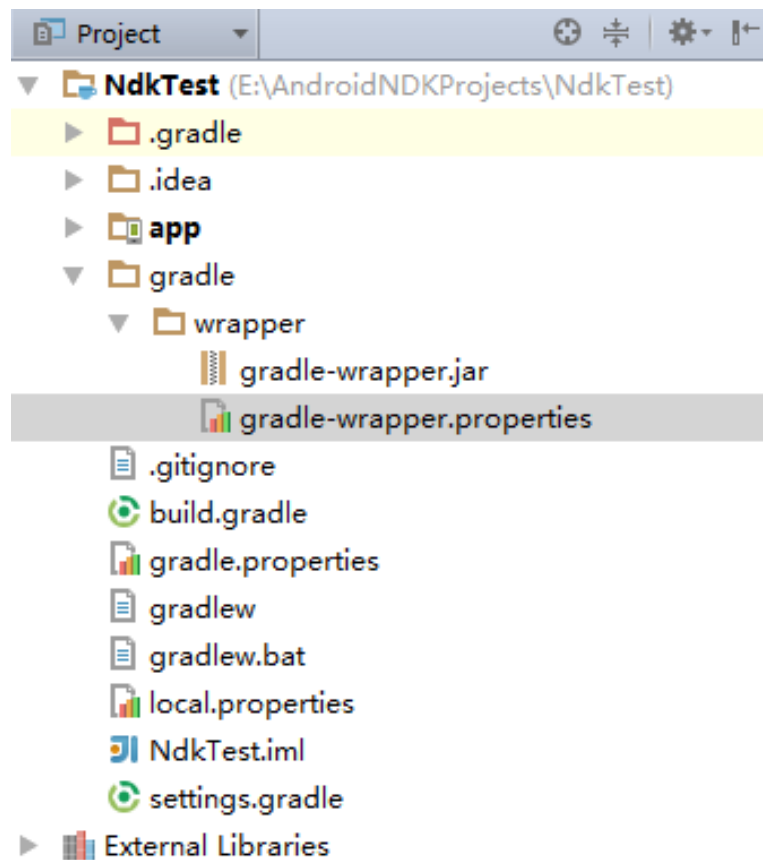
然后是项目路径：

Project location（项目路径）：就是说这个项目的文件装在哪一个文件夹里，这个可以随便设，只要自己记得住。后面有一步选模板，最好选“Empty Activity”，其他模板会给你预备一堆没用的组件。

二、改gradle代码

对于不需要C/C++的项目，这一步完全可以跳过。但是不需要C/C++，也就不必看这篇文章了。

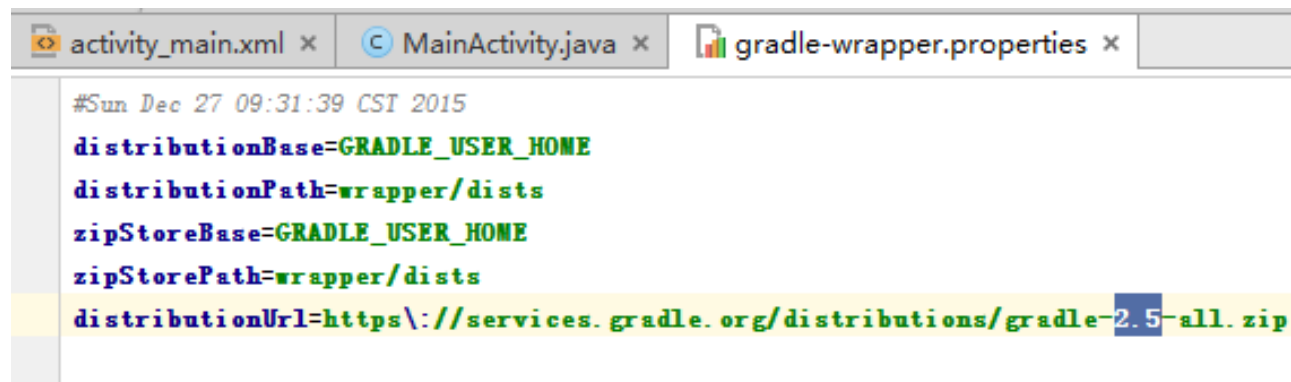
打开项目后，在界面左上方选“Project”，再把目录展开成这样：



上图中选中的“gradle-wrapper.properties”，是马上要修改的。它在目录中的位置，可表示为“gradle/wrapper/gradle-wrapper.properties”。

双击它，就在界面右边看到了它的代码，将“2.4”改成“2.5”，如下图所示。

这么改，是因为目前的NDK只支持gradle2.5，版本高了或低了都不行。



```
#Sun Dec 27 09:31:39 CST 2015
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-2.5-all.zip
```

如法炮制，修改build.gradle，将“:1.3.0”改为“-experimental:0.2.0”。

这是因为目前的NDK需要一个叫“experimental”的插件，它目前的版本是0.2.0。

注意：在现在以及后面的代码修改中，一个字符也不要错，仔细看本文的说明，一个冒号、一个点也不要漏掉，不然调试时报错，你都看不懂（调试不会说“你这里缺了一个分号”，它只会说一句让你丈二和尚摸不着头脑的话，甚至跟真正的错误八竿子也打不着）。

改完之后，立刻会冒出一个***警告，说gradle已经被你改了，你得在项目中更新它。这件事待会儿再做。



```
activity_main.xml x MainActivity.java x gradle-wrapper.properties x NdkTest x

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

接下来修改app/build.gradle（注意，有两个build.gradle，刚才改的是根目录下的，现在改的是app目录下的）。这里要改的地方特别多，先用纯文本标一下（你别复制这段代码，因为可能有些数字和你的项目不匹配，你就照着这个模板改你的代码）。

再次提醒：必要的地方，一个字也别漏掉，一个也别多，哪怕是一个大小写的区别。

apply plugin: 'com.android.model.application' //红色代码是新加的

```
model {
    android {
        compileSdkVersion = 23 //这些数字是软件根据你选择的版本自动写的，不必改
        buildToolsVersion = "23.0.1"
        defaultConfig.with {
            applicationId = "myself.exercise.ndktest" //这是程序包名，用你自己的
            minSdkVersion.apiLevel = 11
            targetSdkVersion.apiLevel = 23
            versionCode = 1
            versionName = "1.0"
        }
        tasks.withType(JavaCompile) {
            sourceCompatibility = JavaVersion.VERSION_1_7
            targetCompatibility = JavaVersion.VERSION_1_7
        }
    }
    android.ndk {
        moduleName = "lb" //这是将来so文件的名称，自己取
    }
    android.buildTypes { //蓝色代码是移动了的
        release {
            minifyEnabled = true
            proguardFiles.add(file("proguard-rules.pro")) //紫色代码是改过了的
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    //testCompile 'junit:junit:4.12' //灰色代码的句子，你看到了就删
    compile 'com.android.support:appcompat-v7:23.0.1'
    //compile 'com.android.support:design:23.0.1'
}
```

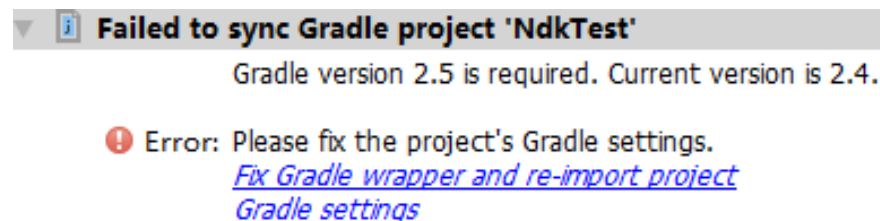
改完了再一行一行检查，这要是错了，调试会痛苦不堪。

说到这里得提一下google的态度，把这么麻烦的事交给用户，他们自己也挺不好意思的，在官网上说，因为NDK还是实验版，不得已才让用户自己改代码，他们会逐渐让这些东西自动化。

三、更新gradle

现在来解决那个***警告的问题。

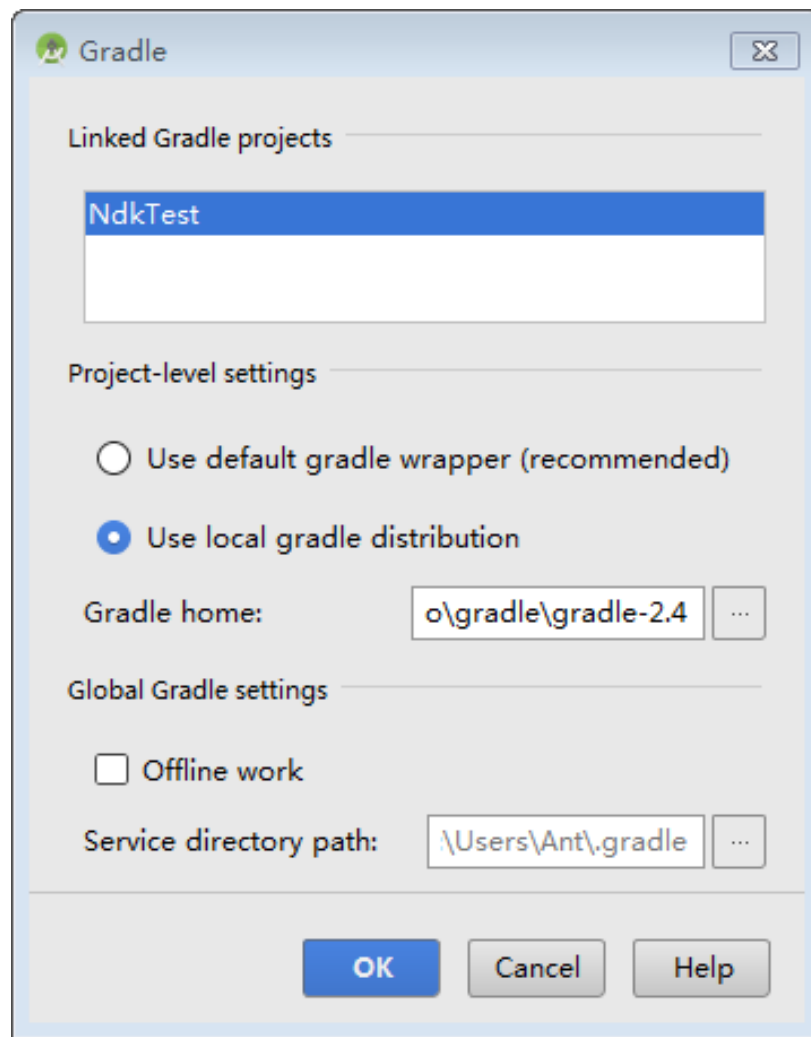
警告右边有一个“Sync Now”链接，点它，故意引发报错：



它是说，现在的gradle是2.4版的，你设置了2.5，就得找到2.5的安装路径。

正如前面所说，目前的NDK只支持gradle2.5，这也就是为什么我们在安装阶段就把2.5放在2.4的旁边。

在上述报错窗口中，点“Gradle settings”，打开如下窗口：



点“gradle-2.4”旁边的“...”按钮，去找你已经安装好的gradle-2.5。之后报错信息没有了，程序会重建gradle。

grayhat

发表于 2016-1-14 23:19 | 来自 51CTO网页

[只看他] 5#



新新人类

帖子 13

精华 0

无忧币 52

个人空间 发短消息

家园好友 他的博客

他的资源

他的课程中心

四、关于实验版插件

还有一个隐藏的家伙一直没有露面，但是跟着NDK一起进入了你的Android Studio，悄悄地进村，打qiang地不要，它就是Experimental Plugin，是C++制作必不可少的。刚才改“build.gradle”已经提到了它，就是“experimental:0.2.0”。

目前Android Studio的C/C++开发离不开这三个家伙（而且对它们的版本有严格限制）：

- NDK（原生开发工具包，Native Develop Kit）r10e版；

- gradle2.5版；

- Experimental Plugin（实验版插件）。

总结一下它们的来历：

- 你单独下载安装了gradle2.5

- 用Android Studio内置的链接下载、由Android Studio自动安装了NDK

- NDK悄悄带来了实验版插件

关于此插件的详情，参阅tools.android.com/tech-docs/new-build-system/gradle-experimental

关于NDK，参阅developer.android.com/intl/zh-cn/ndk/index.html、tools.android.com/tech-docs/android-ndk-preview

五、创建jni文件夹

在app/src/main文件夹上点右键，在弹出菜单中选择“New”、“Folder”、“JNI Folder”，按提示进行。

有一个“Change Folder Location”选项，不需要勾选，因为jni文件夹采用默认的位置（在main文件夹中）就行。

然后main目录下会出现jni文件夹。



微信



关注51CTO博客微信
获得每日精选推荐文章

微信号：blog51cto

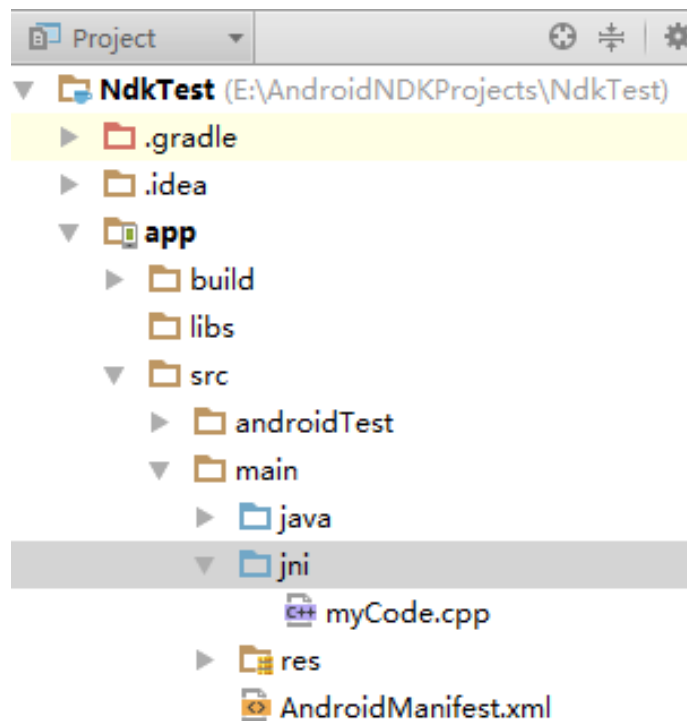
六、创建C++源文件

在jni文件夹上点右键，在弹出菜单中选择“New”、“C/C++ Source File”。

在出现的对话框中，给即将创建的C++文件取名（不要加后缀，软件会自动加上的）。

有一个选项“Create associated header”，不要勾选它，现在不需要创建头文件。

之后，jni文件夹中出现了“myCode.cpp”，我们先不管它，先到Android部分做一些准备工作。



七、建一个演示按钮

这是纯Android操作，可能你已经熟悉了，但初学者需要知道。

编辑主窗口的xml文件（在此范例里是“activity_main.xml”）。如果你在界面上看不到它，就到Project目录的“app/src/main/res/layout/”中找到它，双击它打开编辑窗口。

窗口中有个手机图样，是模拟软件显示的效果，上面有一行字“Hello World”，是AndroidStudio自动加上的显示文字的控件。双击它，会打开一个小窗口，上面有“id”栏，在这里给该控件取名为“textView”（实际上可以随便取，但为了和后面的步骤衔接，就取这个名吧），这是java代码将要引用的。

再添加一个按钮，方法是：在左边的组件列表中找到“Button”，拖到手机图上。

当然你也可以用编辑代码的方式加这个按钮，这里对纯Android操作只说最简单的方法。

双击这个按钮打开小窗口，给它的“id”取名叫“button1”。

然后点上方的“MainActivity.java”选项卡，进入主窗口代码编辑界面，把代码改成这样：

```
package myself.exercise.ndktest;
```

```
import android.app.Activity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    TextView textView; //声明一个标签  
    Button button1; //声明一个按钮
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);
```

```
        textView = (TextView)findViewById(R.id.textView); //在xml中找到标签
```

```
        button1 = (Button)findViewById(R.id.button1); //在xml中找到按钮
```

```
        button1.setOnClickListener(new View.OnClickListener() {
```

@Override

```
        public void onClick(View v) {
```

```
            //点按钮以后发生的事
```

```
        }
```

```
    });
```

```
}
```

```
}
```

新手注意：顶端的这一句：“package myself.exercise.ndktest;”，表明这个项目的包名是“myself.exercise.ndktest”，以后引用包名时，可以到这儿来拷贝。

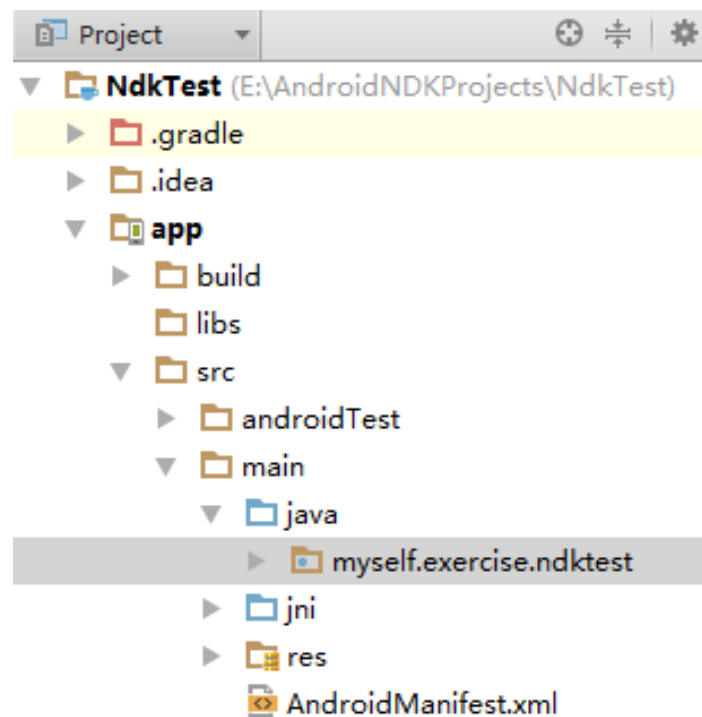
另外，在Android Studio中输入类名（比如“TextView”）时，如果你没写错它却显示为红色，表明Android Studio还没有识别这个名称，上面的“import”还缺点什么，你有两种办法：

1. 重新输入它，等待一个提示框出现，再按Tab键，Android Studio会自动帮你完成输入，把相应的“import”添加到代码中，比如添加“import android.widget.TextView;”，这时就不报错了。

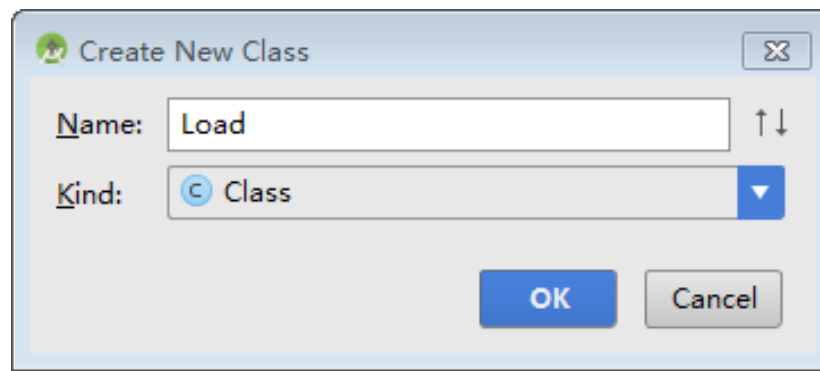
2. 单击它，当提示框出现时，按“Alt + 回车键”。

八、建一个新的java类，用来加载C++库

在“Project”目录中找到“app/src/main/java/myself.exercise.ndktest”：



在这个文件夹上点右键，在弹出菜单中选择“New”、“Java Class”，在弹出窗口中取名“Load”（不要加后缀，软件会自动加上的）：



点“OK”之后，“myself.exercise.ndktest”文件夹中就出现了新的类“Load”（其实它的全名是“Load.java”）。双击它打开它的代码窗口，把代码改成这样：

```
package myself.exercise.ndktest;
```

```
public class Load {  
    static { //载入名为“lb”的C++库  
        System.loadLibrary("lb");  
    }  
    public native int addInt(int a, int b); //调用库里的方法“addInt”，这是计算a和b两个整数相加  
}
```

```
package myself.exercise.ndktest;

public class Load {
    static { //载入名为“lb”的C++库
        System.loadLibrary("lb");
    }

    public native int addInt(int a, int b); //调用库里的方法“addInt”，这是计算a和b两个整数相加
}
```

现在“addInt”是红色的，因为前面的“native”还没有找到下落，以后等native函数（C/C++函数）有了，这里自然就不红了。

现在记住几个关键的名称：

- 即将创建的C++库的名称：lb
- 加载此库的java包名：myself.exercise.ndktest
- 加载此库的java类名：Load
- 从此库中调用的方法：int addInt(int a, int b)（参数是两个整数，返回是整数）

AD 51CTO社区：活动汇总【9个活动正在进行中】 | #论坛扒点档#，随你造！

grayhat

发表于 2016-1-14 23:25 | 来自 51CTO网页

[只看他] 6#

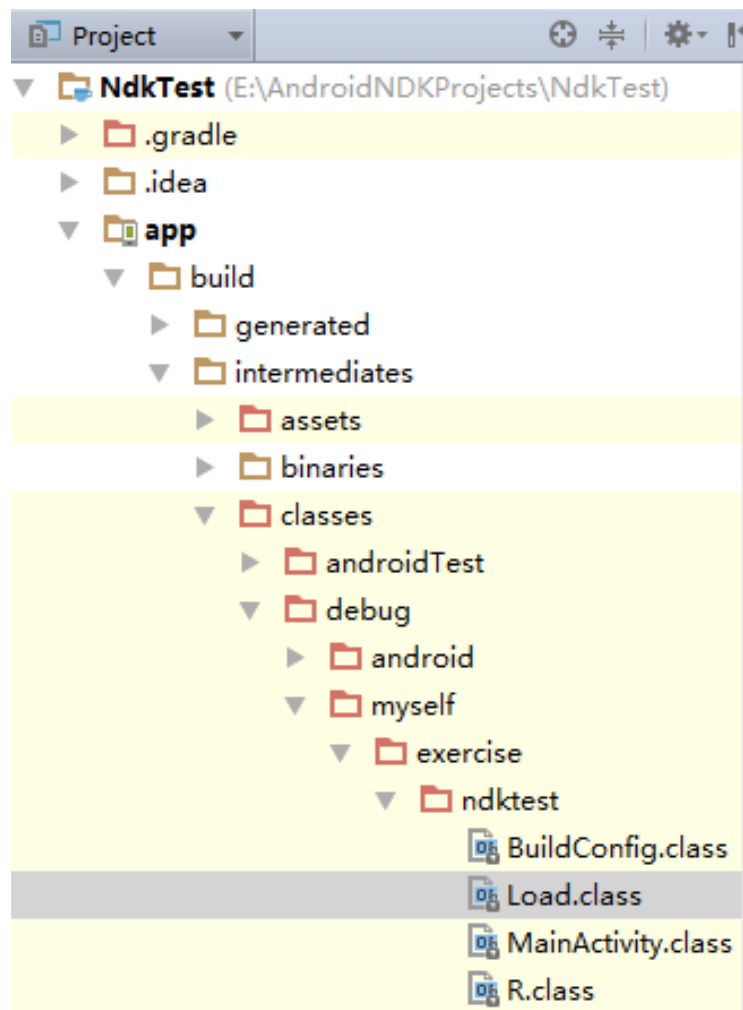


新新人类

帖子 13

九、生成头文件

走一遍主菜单“Build > Make Project”，这一步有没有效果，要看这里：

[个人空间](#) [发短消息](#)[家园好友](#) [他的博客](#)[他的资源](#)[他的课程中心](#)

原来是没有“Load.class”的，但是“Make Project”之后，它就加进来了。前提是“Make Project”要成功，不成功，什么也不会改变。至于不成功的原因，可能是你漏了哪个字符，可能是没有严格按本指南操作，也可能你的软件版本与本文所说的不一样。软件开发是由一堆一堆人为的规矩组成的，不是理论物理，不是纯粹数学，没有真理，我们永远挣扎在别人制定的规矩中，没办法，因为我们没有比尔·盖茨那样的本事。

点界面左下角的“Terminal”标签，打开命令行窗口：

```
Terminal
+ Microsoft Windows [版本 6.1.7601]
X 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

E:\AndroidNDKProjects\NdkTest>
```

在这句话的末尾输入：

```
cd app/build/intermediates/classes/debug
```

按回车键，进入“debug”目录：

```
Terminal
+ Microsoft Windows [版本 6.1.7601]
X 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

E:\AndroidNDKProjects\NdkTest>cd app/build/intermediates/classes/debug

E:\AndroidNDKProjects\NdkTest\app\build\intermediates\classes\debug>
```

接着输入：

```
javah -jni myself.exercise.ndktest.Load
```

按回车键。

Terminal

+ Microsoft Windows [版本 6.1.7601]

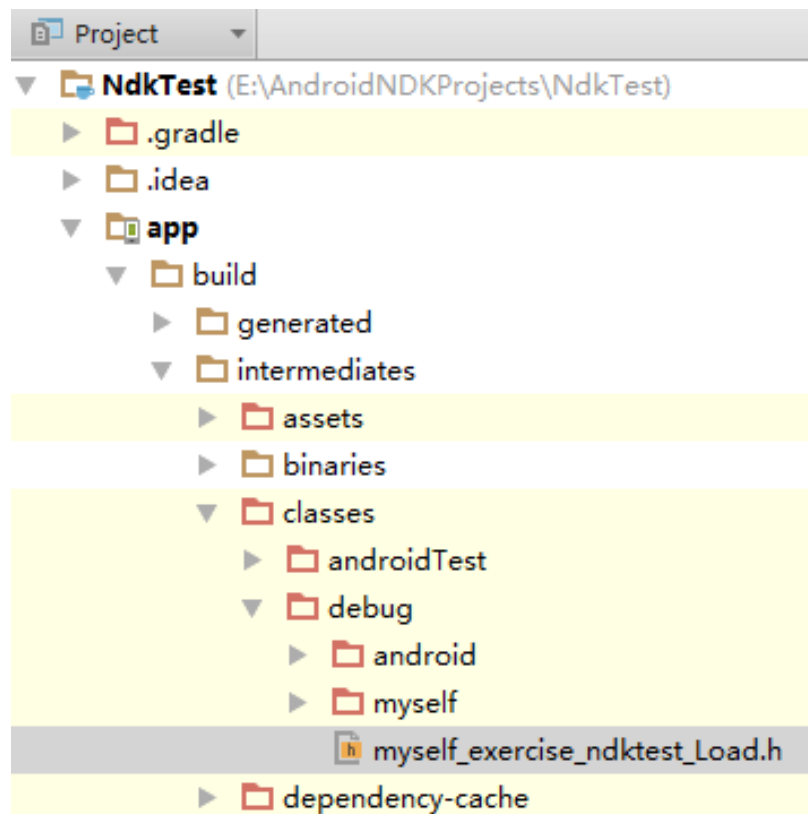
× 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

```
E:\AndroidNdkProjects\NdkTest>cd app/build/intermediates/classes/debug
```

```
E:\AndroidNdkProjects\NdkTest\app\build\intermediates\classes\debug>javah -jni myself.exercise.ndktest.Load
```

```
E:\AndroidNdkProjects\NdkTest\app\build\intermediates\classes\debug>
```

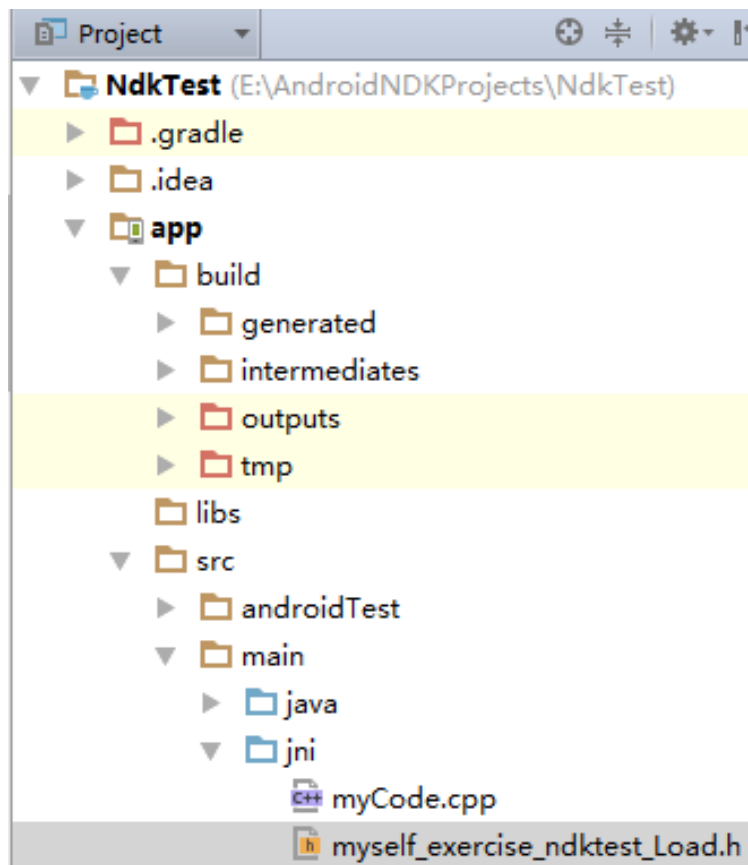
注意：“myself.exercise.ndktest”是包名，“Load”是调用C++库的类的名称，你开发自己的软件时，这些要改的。
要是你没写错，“debug”目录下会有一个头文件生成：



它的后缀是“h”，表明它是C/C++头文件；它的名字仍然由上述包名和类名组成，只不过点变成了横杠。

右键单击此文件，在弹出菜单中选择“Cut”。

然后找到“app/src/main/jni”目录，在jni文件夹上右键单击，在弹出菜单中选择“Paste”，在接下来的对话框中什么也不改直接点“OK”，你会看到头文件已经被转移到了jni文件夹中，与我们先前建立的源文件“myCode.cpp”文件在一起。



双击此文件，可以看到它的代码：

```

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

#ifndef _Included_myself_exercise_ndktest_Load
#define _Included_myself_exercise_ndktest_Load
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      myself_exercise_ndktest_Load
 * Method:     addInt
 * Signature:  (II)I
 */
JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt
    (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif

```

C++的普通语法不解释了，这里特别的一段是：

```

JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt
    (JNIEnv *, jobject, jint, jint);

```

它声明了方法“addInt”，并指定了可以调用它的java包是“myself.exercise.ndktest”、可以调用它的java类是“Load”。“jint”与“JNICALL”之间有一个红色警告，不管它，这是假报错。

上面有一句“extern "C"”，表明此方法是向外输出的，是给java程序调用的。
在这里，我们什么也不用改。

十、编辑源文件

双击“myCode.cpp”打开其代码编辑窗口，写入以下代码：

```
#include "myself_exercise_ndktest_Load.h"
```

```
JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt (JNIEnv *, jobject, jint a, jint b) {  
    return a + b;  
}
```

这是方法“addInt”的实现——两个整数相加，得到一个整数结果。

grayhat

发表于 2016-1-14 23:30 | 来自 [51CTO网页](#)

[只看他] 7#



新新人类

十一、调用库方法

刚才在“Load.java”中加载了库，调用了库方法，现在在主窗口中调用“Load.java”所调用的这个方法。


帖子 [13](#)


精华 [0](#)

无忧币 [52](#)

个人空间  发短消息

 家园好友  他的博客

 他的资源

 他的课程中心


主窗口的代码刚才已经贴过了，现在，在“点按钮以后发生的事”这句话下面添加：

```
Load load = new Load();  
int r = load.addInt(100, 50);  
textView.setText("C++库计算“100+50”的结果是：" + String.valueOf(r));
```

整个过程是：

1. 调试或发布时，“myCode.cpp”将被编译为库“lb”。
2. “Load”加载了库“lb”，调用了库中的方法“addInt”。
3. 主窗口的按钮点击事件，通过“Load”调用这个方法，算出100+50等于多少。

十二、调试

手机连好，点界面上方的调试按钮  。

过一会儿弹出一个窗口，在其中选择已连好的手机，点“OK”，等一会儿到手机上看结果。

不出意外的话，手机上会出现刚做的软件，点其中的按钮会显示C++计算的结果：



grayhat

发表于 2016-1-14 23:34 | 来自 [51CTO网页](#)

[只看他] 8#



新新人类

帖子 [13](#)

精华 [0](#)

无忧币 [52](#)

个人空间 发短消息

家园好友 他的博客

他的资源

他的课程中心

【练习小结】（只说C++的部分）

一、改gradle代码

1. “gradle/wrapper/gradle-wrapper.properties”中，“2.4”改成“2.5”。
2. “build.gradle”中，“:1.3.0”改为“-experimental:0.2.0”。
3. “app/build.gradle”中：

apply plugin: 'com.android.model.application' //红色代码是新加的

```

model {
    android {
        compileSdkVersion = 23 //这些数字是软件根据你选择的版本自动写的，不必改
        buildToolsVersion = "23.0.1"
        defaultConfig.with {
            applicationId = "myself.exercise.ndktest" //这是程序包名，用你自己的
            minSdkVersion.apiLevel = 11
            targetSdkVersion.apiLevel = 23
            versionCode = 1
            versionName = "1.0"
        }
        tasks.withType(JavaCompile) {
            sourceCompatibility = JavaVersion.VERSION_1_7
            targetCompatibility = JavaVersion.VERSION_1_7
        }
    }
    android.ndk {
        moduleName = "lb" //这是将来so文件的名称，自己取
    }
    android.buildTypes { //蓝色代码是移动了的
        release {
            minifyEnabled = true
            proguardFiles.add(file("proguard-rules.pro")) //紫色代码是改过了的
        }
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    //testCompile 'junit:junit:4.12' //灰色代码的句子，你看到了就删
    compile 'com.android.support:appcompat-v7:23.0.1'
    //compile 'com.android.support:design:23.0.1'
}

```

二、更新 **gradle**

“Sync Now”、“Gradle settings”，找gradle-2.5。

三、创建 **jni** 文件夹，在此文件夹中创建 **C++** 源文件 ***.cpp**

四、建一个新的java类，用来加载C++库

代码模板：

```
package myself.exercise.ndktest;
public class Load {
    static {
        System.loadLibrary("lb");
    }
    public native int addInt(int a, int b);
}
```

五、生成头文件*.h

主菜单“Build > Make Project”。

Terminal窗口中：

cd app/build/intermediates/classes/debug 回车

javah -jni myself.exercise.ndktest.Load 回车

把头文件挪到jni文件夹中与源文件*.cpp在一起。

六、编辑源文件*.cpp

代码模板：

```
#include "myself_exercise_ndktest_Load.h"
JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt (JNIEnv *, jobject, jint a, jint b) {
    return a + b;
}
```

七、调用库方法

代码模板：

```
Load load = new Load();
int r = load.addInt(100, 50);
```

grayhat

发表于 2016-1-14 23:38 | 来自 [51CTO网页](#)

[只看他] 9#



新新人类

帖子 [13](#)

精华 [0](#)

无忧币 [52](#)

【更多的jni函数】

刚才处理的只是整数，现在扩展一下。

首先“Load.java”增加了一些函数声明，先在这里声明，以后在C++中实现：

个人空间 发短消息

家园好友 他的博客

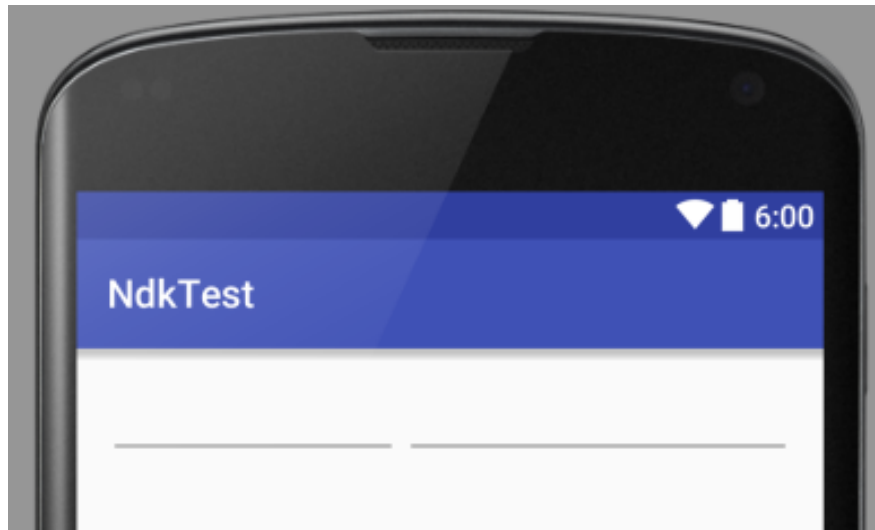
他的资源

他的课程中心

```
package myself.exercise.ndktest;
```

```
public class Load {  
    static {  
        System.loadLibrary("lb");  
    }  
    public native int addInt(int a, int b); //输入整数，输出整数  
    public native double mulDouble(double a, double b); //输入实数，输出实数  
    public native boolean bigger(float a, float b); //输入float型实数，输出布尔值  
    public native String addString(String a, String b); //输入字符串，输出字符串  
    public native int[] intArray(int[] a); //输入整数数组，输出整数数组  
    public native double[] doubleArray(double[] a); //输入实数数组，输出实数数组  
    public native String[] stringArray(String[] a); //输入字符串数组，输出字符串数组  
}
```

然后主窗口增加了一些组件，用来测试这些函数：





为了省事，你可以直接把下述代码拷到“activity_main.xml”中去：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
```

```
<TextView android:text="Hello World!" android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="111dp" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="addInt"
    android:id="@+id/button1"
    android:layout_marginTop="41dp"
    android:layout_below="@+id/textView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
```

```
<EditText
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:id="@+id/editText1"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:width="150dp" />
```

<EditText

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/editText2"
android:width="200dp"
android:layout_alignTop="@+id/editText1"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true" />
```

<Button

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="mulDouble"
android:id="@+id/button2"
android:layout_above="@+id/button3"
android:layout_alignLeft="@+id/button4"
android:layout_alignStart="@+id/button4" />
```

<Button

```
android:layout_width="wrap_content"
```



```
android:layout_height="wrap_content"
android:text="intArray"
android:id="@+id/button3"
android:layout_below="@+id/button1"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true" />
```

<Button

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="doubleArray"
android:id="@+id/button4"
android:layout_below="@+id/button2"
android:layout_centerHorizontal="true" />
```

<Button

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="stringArray"
android:id="@+id/button5"
android:layout_below="@+id/button3"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true" />
```

<Button

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:text="addString"  
android:id="@+id/button6"  
android:layout_below="@+id/button5"  
android:layout_alignParentLeft="true"  
android:layout_alignParentStart="true" />
```

</RelativeLayout>

然后重新生成头文件（因为新增函数的声明需要写在头文件中，我们自己写，不如让Android Studio替我们写），具体做法是：

再走一遍“Build > Make Project”。

在Terminal窗口中再来一遍：

（如果没有进入“debug”目录就）cd app/build/intermediates/classes/debug 回车
javah -jni myself.exercise.ndktest.Load 回车

“debug”目录又冒出了一个头文件“myself_exercise_ndktest_Load.h”，和先前做的头文件名字一样。

把它挪到jni文件夹中，覆盖原来那个头文件。

新的头文件中已经包含了新方法的C++声明：

This file has been added after the last project sync with Gradle. Please sync the project again for the NDK support to work properly.

[Sync Now](#)

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

#ifndef _Included_myself_exercise_ndktest_Load
#define _Included_myself_exercise_ndktest_Load
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     myself_exercise_ndktest_Load
 * Method:   addInt
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt
(JNIEnv *, jobject, jint, jint);

/*
 * Class:     myself_exercise_ndktest_Load
 * Method:   mulDouble
 * Signature: (DD)D
 */
JNIEXPORT jdouble JNICALL Java_myself_exercise_ndktest_Load_mulDouble
(JNIEnv *, jobject, jdouble, jdouble);

/*
 * Class:     myself_exercise_ndktest_Load
 * Method:   bigger
 * Signature: (FF)Z
 */
```

上面又出现了***警告，这是又需要合成gradle了，点右边的“Sync Now”即可。

对头文件本身，什么都不用改。

要改的是源文件“myCode.cpp”，改成这样：

```
#include "myself_exercise_ndktest_Load.h"
#include <string.h>
```

```

JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt (JNIEnv *, jobject, jint a, jint b) {
    return a + b; //输入整数，输出整数
}

JNIEXPORT jdouble JNICALL Java_myself_exercise_ndktest_Load_mulDouble (JNIEnv *, jobject, jdouble a, jdouble
b) {
    return a * b; //输入实数，输出实数
}

JNIEXPORT jboolean JNICALL Java_myself_exercise_ndktest_Load_bigger (JNIEnv *, jobject, jfloat a, jfloat b) {
    return a > b; //输入float型实数，输出布尔值，判断a是否大于b
}

char * JstringToCstr(JNIEnv * env, jstring jstr) { //jstring转换为C++的字符数组指针
    char * pChar = NULL;
    jclass classString = env->FindClass("java/lang/String");
    jstring code = env->NewStringUTF("GB2312");
    jmethodID id = env->GetMethodID(classString, "getBytes", "(Ljava/lang/String;)[B");
    jbyteArray arr = (jbyteArray)env->CallObjectMethod(jstr, id, code);
    jsize size = env->GetArrayLength(arr);
    jbyte * bt = env->GetByteArrayElements(arr, JNI_FALSE);
    if(size > 0) {
        pChar = (char*)malloc(size + 1);
        memcpy(pChar, bt, size);
        pChar[size] = 0;
    }
    env->ReleaseByteArrayElements(arr, bt, 0);
    return pChar;
}

JNIEXPORT jstring JNICALL Java_myself_exercise_ndktest_Load_addString (JNIEnv * env, jobject, jstring a, jstring
b) {
    char * pA = JstringToCstr(env, a); //取得a的C++指针

```

```

char * pB = JstringToCstr(env, b); //取得b的C++指针
return env->NewStringUTF(strcat(pA, pB)); //输出a+b
}

JNIEXPORT jintArray JNICALL Java_myself_exercise_ndktest_Load_intArray (JNIEnv * env, jobject, jintArray a) {
    //输入整数数组，将其每个元素加10后输出
    //输入值为a，输出值为b
    int N = env->GetArrayLength(a); //获取a的元素个数
    jint * pA = env->GetIntArrayElements(a, NULL); //获取a的指针
    jintArray b = env->NewIntArray(N); //创建数组b，长度为N
    jint * pB = env->GetIntArrayElements(b, NULL); //获取b的指针
    for (int i = 0; i < N; i++) pB = pA + 10; //把a的每个元素加10，赋值给b中对应的元素
    /*//另一种方法
    env->SetIntArrayRegion(b, 0, N, pA); //b是a的复制品
    for (int j = 0; j < N; j++) pB[j] += 10; //b的每个元素加10
    */
    env->ReleaseIntArrayElements(a, pA, 0); //释放a的内存
    env->ReleaseIntArrayElements(b, pB, 0); //释放b的内存
    return b; //输出b
}

JNIEXPORT jdoubleArray JNICALL Java_myself_exercise_ndktest_Load_doubleArray (JNIEnv * env, jobject,
jdoubleArray a) {
    //输入实数数组，将其每个元素乘2后输出
    //输入值为a，输出值为b
    int N = env->GetArrayLength(a); //获取a的元素个数
    jdouble * pA = env->GetDoubleArrayElements(a, NULL); //获取a的指针
    jdoubleArray b = env->NewDoubleArray(N); //创建数组b，长度为N
    jdouble * pB = env->GetDoubleArrayElements(b, NULL); //获取b的指针
    for (int i = 0; i < N; i++) pB = pA * 2; //把a的每个元素乘2，赋值给b中对应的元素
    /*//另一种方法
    env->SetDoubleArrayRegion(b, 0, N, pA); //b是a的复制品

```

```

        for (int j = 0; j < N; j++) pB[j] *= 2; //b的每个元素乘2
    */
    env->ReleaseDoubleArrayElements(a, pA, 0); //释放a的内存
    env->ReleaseDoubleArrayElements(b, pB, 0); //释放b的内存
    return b; //输出b
}

JNIEXPORT jobjectArray JNICALL Java_myself_exercise_ndktest_Load_stringArray (JNIEnv * env, jobject,
jobjectArray a) {
    //输入字符串数组，颠倒顺序后输出
    //输入值为a，输出值为b
    int N = env->GetArrayLength(a); //获取a的元素个数
    jobjectArray b = env->NewObjectArray(N, env->FindClass("java/lang/String"), env->NewStringUTF("")); //创建数
组b，长度为N
    for (int i = 0; i < N; i++) { //对于a中的每个元素
        jstring ai = (jstring)env->GetObjectArrayElement(a, i); //读出这个元素的值
        env->SetObjectArrayElement(b, N - 1 - i, ai); //赋值给b中倒序对应的元素
        env->DeleteLocalRef(ai); //释放内存
    }
    return b; //输出b
}

```

再把主窗口代码改成这样：

```
package myself.exercise.ndktest;
```

```
import android.app.Activity;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    Load load = new Load();
    EditText editText1; EditText editText2;
    TextView textView;
    Button button1; Button button2; Button button3;
    Button button4; Button button5; Button button6;
```

[@Override](#)

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

```
    editText1 = (EditText)findViewById(R.id.editText1); //左边的文字输入框，以下简称“左框”
    editText2 = (EditText)findViewById(R.id.editText2); //右边的文字输入框，以下简称“右框”
    textView = (TextView)findViewById(R.id.textView);
    button1 = (Button)findViewById(R.id.button1);
    button1.setOnClickListener(new View.OnClickListener() {
```

[@Override](#)

```
    public void onClick(View v) {
        //按钮“addInt”用来计算左框里的整数和右框里的整数相加
```

```

        //你要是不输入整数，它就当你输入了“0”
        int a = 0;
        try { a = Integer.parseInt(String.valueOf(editText1.getText())); }
        catch (NumberFormatException e) {}
        int b = 0;
        try { b = Integer.parseInt(String.valueOf(editText2.getText())); }
        catch (NumberFormatException e) {}
        int r = load.addInt(a, b);
        textView.setText("C++库计算" + a + "+" + b + "的结果是：" + r);
    }
});
button2 = (Button)findViewById(R.id.button2);
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //按钮“mulDouble”用来计算左框里的实数和右框里的实数相乘
        //你要是不输入数字，它就当你输入了“0”
        double a = 0;
        try { a = Double.parseDouble(String.valueOf(editText1.getText())); }
        catch (NumberFormatException e) {}
        double b = 0;
        try { b = Double.parseDouble(String.valueOf(editText2.getText())); }
        catch (NumberFormatException e) {}
        double r = load.mulDouble(a, b);
        textView.setText("C++库计算" + a + "*" + b + "的结果是：" + r);
    }
});
button3 = (Button)findViewById(R.id.button3);
button3.setOnClickListener(new View.OnClickListener() {
    @Override

```



```

public void onClick(View v) {
    //按钮“intArray”用来计算左框里的整数数组每个元素加10会变成什么，结果放在右框里
    //输入数组时，元素之间用英文逗号隔开，比如“1,2,3”
    //对于每个元素，你要是不输入整数，它就当你输入了“0”
    //注意不要用中文逗号，不然它无法识别
    String str = String.valueOf(editText1.getText());
    String[] strArray;
    if (str.contains(",")) strArray = str.split(",");
    else strArray = new String[] { str };
    int N = strArray.length;
    int[] array = new int[N];
    for (int i = 0; i < N; i++) {
        int t = 0;
        try { t = Integer.parseInt(strArray[i]); }
        catch (NumberFormatException e) {}
        array[i] = t;
    }
    int[] newArray = load.intArray(array);
    str = "";
    for (int i = 0; i < N; i++) {
        if (i < N - 1) str += String.valueOf(newArray[i]) + ",";
        else str += String.valueOf(newArray[i]);
    }
    editText2.setText(str);
    textView.setText("C++库用左边的数组计算，结果在右边");
}
});
button4 = (Button)findViewById(R.id.button4);
button4.setOnClickListener(new View.OnClickListener() {
    @Override

```

```
public void onClick(View v) {
```

```
    //按钮“doubleArray”用来计算左框里的实数数组每个元素乘2会变成什么，结果放在右框里
```

```
    //输入数组时，元素之间用英文逗号隔开，比如“2.5,3.14,8”
```

```
    //对于每个元素，你要是不输入数字，它就当你输入了“0”
```

```
    //注意不要用中文逗号，不然它无法识别
```

```
    String str = String.valueOf(editText1.getText());
```

```
    String[] strArray;
```

```
    if (str.contains(",")) strArray = str.split(",");
```

```
    else strArray = new String[] { str };
```

```
    int N = strArray.length;
```

```
    double[] array = new double[N];
```

```
    for (int i = 0; i < N; i++) {
```

```
        double t = 0;
```

```
        try { t = Double.parseDouble(strArray[i]); }
```

```
        catch (NumberFormatException e) { }
```

```
        array[i] = t;
```

```
    }
```

```
    double[] newArray = load.doubleArray(array);
```

```
    str = "";
```

```
    for (int i = 0; i < N; i++) {
```

```
        if (i < N - 1) str += String.valueOf(newArray[i]) + ",";
```

```
        else str += String.valueOf(newArray[i]);
```

```
    }
```

```
    editText2.setText(str);
```

```
    textView.setText("C++库用左边的数组计算，结果在右边");
```

```
}
```

```
});
```

```
button5 = (Button)findViewById(R.id.button5);
```

```
button5.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```

public void onClick(View v) {
    //按钮“stringArray”用来把左框里的字符串数组颠倒顺序，结果放在右框里
    //输入数组时，元素之间用英文逗号隔开，比如“China,USA,England”
    //注意不要用中文逗号，不然它把这个逗号当成一个字符，与左右字符串相连成为数组中的一个元素
    String str = String.valueOf(editText1.getText());
    String[] strArray;
    if (str.contains(",")) strArray = str.split(",");
    else strArray = new String[] { str };
    int N = strArray.length;
    String[] newArray = load.stringArray(strArray);
    str = "";
    for (int i = 0; i < N; i++) {
        if (i < N - 1) str += newArray + ",";
        else str += newArray;
    }
    editText2.setText(str);
    textView.setText("C++库颠倒了左边数组的顺序，结果在右边");
}
});

button6 = (Button)findViewById(R.id.button6);
button6.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //按钮“addString”用来把左框里的字符串与右框里的字符串相加
        String a = String.valueOf(editText1.getText());
        String b = String.valueOf(editText2.getText());
        String r = load.addString(a, b);
        textView.setText("C++库计算" + a + "+" + b + "的结果是：" + r);
    }
});

```

```
}  
}
```

好了，调试。

比如我在左边输入“1,2,3,4”，点“intArray”按钮，右边就会是“11,12,13,14”，这是C++处理数组的结果，还可以试其他按钮，详细情况在主窗口代码的注释中。



addInt

intDouble

intArray

doubleArray

stringArray

addString

1

2

3

4

5

6

7

8

9

0

,

.

/

...

:

;

?

!

'

"

◀ 1/4 ▶

@

&

^

~

(

)

*

#



本帖最后由 grayhat 于 2016-1-15 00:40 编辑

grayhat

发表于 2016-1-14 23:42 | 来自 [51CTO网页](#)

[只看他] 10#



新新人类

帖子 [13](#)

精华 [0](#)

无忧币 [52](#)

个人空间 发短消息

家园好友 他的博客

他的资源

他的课程中心

【更多更多的jni函数】

基本的函数，语句是相似的，比如取得数组指针，对于int数组，这个函数是GetIntArrayElements，对于byte数组，就是GetByteArrayElements，把Int改成Byte就行了。下面用byte总结一下最常用的函数：

```
private jobjectArray arrToArrArr(JNIEnv * env, jbyteArray arr) {
    int N = env->GetArrayLength(arr); //取得数组长度
    jbyte * pArr = env->GetByteArrayElements(arr, NULL); //取得数组指针
    int M = 16;
    jobjectArray arrArr = env->NewObjectArray(M, env->FindClass("[B"), NULL); //创建二维数组
    for (int i2 = 0; i2 < M; i2++) {
        jbyteArray arrCopy = env->NewByteArray(N); //创建一维数组
```

```

        jbyte * pArrCopy = env->GetByteArrayElements(arrCopy, NULL); //取得相应指针
        for (int j = 0; j < N; j++) pArrCopy[j] = pArr[j] + 1; //一维数组元素操作
        env->SetObjectArrayElement(arrArr, i2, arrCopy); //二维数组元素操作
        env->ReleaseByteArrayElements(arrCopy, pArrCopy, 0); //释放一维数组指针
        env->DeleteLocalRef(arrCopy); //释放对它的引用
    }
    jobject obj = env->GetObjectArrayElement(arrArr, 0); //取二维数组的元素
    jbyteArray newArr = (jbyteArray)obj; //强制转换
    int thisN = env->GetArrayLength(newArr);
    jbyte * pNewArr = env->GetByteArrayElements(newArr, NULL); //取指针
    for (int i3 = 0; i3 < thisN; i3++) pNewArr[i3] += 2; //元素操作
    env->SetObjectArrayElement(arrArr, M - 1, newArr); //用修改后的一维数组替代二维数组的最后一行
    env->ReleaseByteArrayElements(newArr, pNewArr, 0); //释放这个一维数组的指针
    env->DeleteLocalRef(newArr); //释放对它的引用
    env->ReleaseByteArrayElements(arr, pArr, 0); //释放最初那个一维数组的指针
    return arrArr;
}

JNIEXPORT jobjectArray JNICALL Java_myself_exercise_ndktest_Load_arrToArrArr(JNIEnv * env, jobject,
jbyteArray arr) {
    return arrToArrArr(env, arr);
}

```

jni中的基本数据类型有：

jboolean：对应于C++的bool、java的boolean

jbyte：对应于C++和java的byte

jchar：对应于C++和java的char

jshort：对应于C++和java的short

jint：对应于C++和java的int

jlong：对应于C++和java的long

jfloat：对应于C++和java的float

jdouble：对应于C++和java的double

带“JNIEXPORT”的外联语句应该使用“j”字头数据类型，而在jni内部，对于基本数据类型，一般来说既可以用jni式的，也可以用C++式的，比如定义一个名叫“flag”的变量，可以用“jboolean flag = false”，也可以用“bool flag = false”，涉及到整数，可以用“jint”，也可以用“int”。但byte是个例外，jni不支持“byte”这个关键词，只能用“jbyte”。

数组类型

有jbooleanArray、jbyteArray、jcharArray、jshortArray、jintArray、jlongArray、jfloatArray、jdoubleArray，实际上就是每个基本类型都有一个Array。它们的函数写法也是类似的，比如上面那段代码，把“byte”统一替换成“int”，把“Byte”统一替换成“Int”，就是jint型的函数了（不过涉及到二维数组时，“FindClass("[B]")”要改成“FindClass("[I]")”）。

jni数组可以写成C++指针，比如：

```
jcharArray chArr = env->NewCharArray(10);
jchar * pChArr = env->GetCharArrayElements(chArr, NULL);
jintArray intArr = env->NewIntArray(10);
jint * pIntArr = env->GetIntArrayElements(intArr, NULL);
for (int i = 0; i < 10; i++) {
    pChArr = 65 + i;
    pIntArr = pChArr + 100;
}
env->ReleaseCharArrayElements(chArr, pChArr, 0);
env->ReleaseIntArrayElements(intArr, pIntArr, 0);
```

可以写成C++式的：


```
char chArr[10];
int intArr[10];
for (int i = 0; i < 10; i++) {
    *(chArr + i) = 65 + i;
    *(intArr + i) = *(chArr + i) + 100;
}
free(chArr);
free(intArr);
```

但不能、也不可能写成java式的，像“char[] chArr = new char[10]”这样的句子进去以后直接就报错了，就变红了。
字符串是比较特别的类型，按理说它是字符数组，但它的函数写法与众不同：

```
jstring str = env->NewStringUTF("abcde"); //创建字符串
const jchar * pJchars = env->GetStringChars(str, NULL); //获得Unicode编码的字符串指针
const char * pChars = env->GetStringUTFChars(str, NULL); //获得UTF-8编码的字符串指针
const jbyte * pBytes = env->GetStringUTFChars(str, NULL); //获得与UTF-8编码的字符串等价的byte数组指针
int N = env->GetStringLength(str); //获得Unicode编码的字符串长度
int start = 0; int n = N - 1;
jchar * jchBuf;
env->GetStringRegion(str, start, n, jchBuf); //从序号为start的字符开始，把n个字符复制给Unicode编码的jchar数组
int utfN = env->GetStringUTFLength(str); //获得UTF-8编码的字符串长度
char * chBuf;
env->GetStringUTFRegion(str, start, n, chBuf); //从序号为start的字符开始，把n个字符复制给UTF-8编码的char数组
const jchar * cr = env->GetStringCritical(str, NULL); //获得编码格式的字符串指针
env->ReleaseStringChars(str, pJchars); //由上述函数产生的各种指针的释放
env->ReleaseStringUTFChars(str, pChars);
env->ReleaseStringUTFChars(str, pBytes);
env->ReleaseStringChars(str, jchBuf);
```

```
env->ReleaseStringUTFChars(str, chBuf);
```

```
env->ReleaseStringCritical(str, cr);
```

并不能用string来代替jstring，因为jni中根本就没有string这个写法。但对于char*指针，可以用C++的函数，比如strcpy。

一个简单的例子，注册校验中常用的，比较两个字符串是否相同：

```
bool sameJstring(JNIEnv *env, jstring a, jstring b) {  
    if (a == NULL) {  
        if (b == NULL) return true;  
        else return false;  
    } else {  
        if (b == NULL) return false;  
        else {  
            int aN = env->GetStringUTFLength(a);  
            int bN = env->GetStringUTFLength(b);  
            if (aN <= 0) {  
                if (bN <= 0) return true;  
                else return false;  
            } else {  
                if (bN <= 0) return false;  
                else {  
                    if (aN != bN) return false;  
                    else {  
                        const char * pA = env->GetStringUTFChars(a, NULL); //得到这两个字符串的指针  
                        const char * pB = env->GetStringUTFChars(b, NULL);  
                        bool result = true;  
                        for (int i = 0; i < aN; i++) {
```

```
    if (pA != pB) { //指到每个位置的字符，进行比较
        result = false;
        break;
    }
}
env->ReleaseStringUTFChars(a, pA);
env->ReleaseStringUTFChars(b, pB);
return result;
}
}
}
}
```

更多的例子到网上搜吧，写不下了。

网上很多例子是老版本NDK的，与Eclipse合作的，有很多这样的句子：“(*env)->函数名(env, 其他参数)”。这个，到了本文所说的NDK版本中，统一替换成“env->函数名(其他参数)”，就是把前面那个“env”两边的括号和星号去掉，把参数中那个“env,”去掉。

有时候这样还报错，你可以考虑是不是表达式左右的类型不匹配，试试强制转换，比如老版本的“`JSONArray arr = (*env)->GetObjectField(env, packageInfo, fieldID);`”改成“`JSONArray arr = env->GetObjectField(packageInfo, fieldID_signatures);`”之后还报错，就试试“`JSONArray arr = (JSONArray)env->GetObjectField(packageInfo, fieldID_signatures);`”。

但有些老句子是彻底不行了，比如“`int iReturn = __system_property_get(key, value);`”调试时报错“Error:(87) undefined reference to `__system_property_get'”，这是因为NDK10已经把“`__system_property_get`”废了，这没办法。

有一类例子，几乎从来没人把它说清楚，就是引用java的Context。

比如说查签名，这得通过java查，要把java的环境（Context）导进来，才知道里面的东西比如签名是什么样子的。在网上的例子中，这类函数都有一个参数“object context”，有时候写成“object obj”之类的，但是，从来就没有人说清楚这

个context或obj从哪来，很容易让人误解为是这样：

```
jstring getSignature(JNIEnv *env, jobject context) {
    jstring currentSignature;
    //获取签名
    return currentSignature;
}

Jboolean compareSignatures(JNIEnv *env, jstring legalSignature, jstring currentSignature) {
    jboolean result;
    //比较正确的签名和当前的签名
    return result;
}

JNIEXPORT Jboolean JNICALL Java_myself_exercise_ndktest_Load_compareSignatures(JNIEnv * env, jobject
context) {
    return compareSignatures(env, context); //以为上面的context是从这儿来的，错！
}
```

其实，这个context需要一个专门的函数，从java中调用：

```
/*获取当前应用的Context*/
jobject getContext(JNIEnv *env) {
    jclass activityThread = env->FindClass("android/app/ActivityThread");
    jmethodID currentActivityThread = env->GetStaticMethodID(activityThread, "currentActivityThread",
"()Landroid/app/ActivityThread;");
    jobject at = env->CallStaticObjectMethod(activityThread, currentActivityThread);
    jmethodID getApplication = env->GetMethodID(activityThread, "getApplication", "()Landroid/app/Application;");
    jobject context = env->CallObjectMethod(at, getApplication);
}
```

```

        return context;
    }
    /*获取当前应用签名的HashCode*/
    int getSignatureHashCode(JNIEnv *env) {
        jobject context = getContext(env); //context是这么来的！
        jclass contextClass = env->GetObjectClass(context);
        jmethodID methodID_getPackageManager = env->GetMethodID(contextClass, "getPackageManager",
            "()Landroid/content/pm/PackageManager;");
        jobject packageManager = env->CallObjectMethod(context, methodID_getPackageManager);
        jclass packageManagerClass = env->GetObjectClass(packageManager);
        jmethodID methodID_getPackageInfo = env->GetMethodID(packageManagerClass, "getPackageInfo",
            "(Ljava/lang/String;I)Landroid/content/pm/PackageInfo;");
        jmethodID methodID_getPackageName = env->GetMethodID(contextClass, "getPackageName",
            "()Ljava/lang/String;");
        jstring packageName = (jstring)env->CallObjectMethod(context, methodID_getPackageName);
        jobject packageInfo = env->CallObjectMethod(packageManager, methodID_getPackageInfo, packageName,
            64);
        jclass packageInfoClass = env->GetObjectClass(packageInfo);
        jfieldID fieldID_signatures = env->GetFieldID(packageInfoClass, "signatures",
            "[Landroid/content/pm/Signature;");
        jobjectArray signature_arr = (jobjectArray)env->GetObjectField(packageInfo, fieldID_signatures);
        jobject signature = env->GetObjectArrayElement(signature_arr, 0);
        jclass signatureClass = env->GetObjectClass(signature);
        jmethodID methodID_hashcode = env->GetMethodID(signatureClass, "hashCode", "()I");
        jint hashCode = env->CallIntMethod(signature, methodID_hashcode);
        return hashCode;
    }
    /*获取当前应用签名的jstring*/
    jstring getSignature(JNIEnv *env) {
        jobject context = getContext(env); //context是这么来的！

```

```

jclass contextClass = env->GetObjectClass(context);
jmethodID methodID_getPackageManager = env->GetMethodID(contextClass, "getPackageManager",
"()Landroid/content/pm/PackageManager;");
jobject packageManager = env->CallObjectMethod(context, methodID_getPackageManager);
jclass packageManagerClass = env->GetObjectClass(packageManager);
jmethodID methodID_getPackageInfo = env->GetMethodID(packageManagerClass, "getPackageInfo",
"(Ljava/lang/String;I)Landroid/content/pm/PackageInfo;");
jmethodID methodID_getPackageName = env->GetMethodID(contextClass, "getPackageName",
"()Ljava/lang/String;");
jstring packageName = (jstring)env->CallObjectMethod(context, methodID_getPackageName);
jobject packageInfo = env->CallObjectMethod(packageManager, methodID_getPackageInfo, packageName,
64);
jclass packageInfoClass = env->GetObjectClass(packageInfo);
jfieldID fieldID_signatures = env->GetFieldID(packageInfoClass, "signatures",
"[Landroid/content/pm/Signature;");
jobjectArray signature_arr = (jobjectArray)env->GetObjectField(packageInfo, fieldID_signatures);
jobject signature = env->GetObjectArrayElement(signature_arr, 0);
jclass signatureClass = env->GetObjectClass(signature);
jmethodID methodID_toCharsString = env->GetMethodID(signatureClass, "toCharsString",
"()Ljava/lang/String;");
return (jstring)env->CallObjectMethod(signature, methodID_toCharsString);
}
/*比较两个jstring是否相同*/
bool sameJstring(JNIEnv *env, jstring a, jstring b) {
    if (a == NULL) {
        if (b == NULL) return true;
        else return false;
    } else {
        if (b == NULL) return false;
        else {

```



```
///比对整个签名
jstring signature = getSignature(env);
jstring legalSignature = env->NewStringUTF("把这个字符串替换成你自己程序的正确签名");
return sameJstring(env, signature, legalSignature);
///只比对签名HashCode
/*jint hashCode = getSignatureHashCode(env);
jint legalHashCode = 0; //把这个“0”替换成你自己程序的正确签名的HashCode
return hashCode == legalHashCode;*/
}
```

/*JNIEXPORT 那一段就不要了，比较的结果不要输出给java，就在jni内部使用。

而且，在实际应用中，简单地返回一个jboolean是不行的，让人找到改成true就破解了。即使在jni内部，变量也可以被修改。能增加破解难度的办法是把校验过程包在一个加密的区块中，结果不外露，向外输出的只是一些被它控制的变量，这些变量的值是不固定的、有无限的取值范围可以让程序正常运转、又有无限的取值范围让程序混乱，因此别人无法通过修改这些变量来达到正版，他破解的唯一办法是把这个区块的秘钥找到。

还有，这些函数的名称，不要用傻乎乎的“getSignature”之类，要用谁也看不懂的词，代码中也不要出现“感谢你使用正版”之类不打自招的字符串，即使区块已经加密，也要保持良好的习惯。

有人也许要问：区块加密之后肯定要重新签名，签名变了就得回jni里面修改，修改过后又得重新加密，加密过后签名又变了……别说了，下载一个可以调用作者jks的签名工具就行了（比如360加固保的签名工具），签名会恢复原状的*/

grayhat

发表于 2016-1-14 23:49 | 来自 [51CTO网页](#)

[只看他] 11#

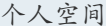


【jni的注意事项】

最重要的是释放内存。jni不像java那样保姆，不会帮你打扫房间的，垃圾要自己扫。

新新人类



帖子 [13](#)
精华 [0](#)
无忧币 [52](#)

 发短消息
 他的博客
 他的课程中心

基本数据类型不用扫，但引用类型（jstring、带“array”的）用过之后必须扫。不扫的结果是什么呢，不像在家里一样躺在垃圾堆里还能睡着，jni循环几轮就可能死给你看。

清扫垃圾大致的规律是：有指针就要扫。因此可以这样：写完一段代码，从上往下搜*符号，有一个算一个都在用完释放。对于jni指针（jintArray之类的），释放语句都是带“Release”的（如env->ReleaseIntArrayElements）；对于纯C++指针，用free来释放，前面都举过例子。

特别提醒：用完才释放，别没用完就把人家给灭了那就查不出什么来了。

其实jni非常脆弱，即使释放，循环次数多了也吃不消。你会遇到这种情况：某个复杂的函数，里面有N多循环，动用了N多指针，你打算从0算到1000，在手机上一调试，到100就死球了，甚至到20就跳出“已停止”的讣告，这还算好的，要是按钮按下后永远起不来直到人工重启，那才气人。这种情况很可能是超负荷了，那就小批小批地传给jni计算，它有个特点，执行完java传来的一个任务会清理一次内存，比代码中的“Release”清理得彻底，所以你让它多和java对话，内存就“时时勤拂拭，勿使惹尘埃”了。

但也有可能是代码本身的问题，这在jni里很难查。Jni打不了断点，调试的劳动量可想而知。那就不要在jni里写太多的东西，当然我们也不指望把整个软件都搬到jni里去。还有一个好习惯，代码先在Android里调试，逻辑上没问题了再改成jni格式搬进cpp文件里，这时候再出问题就是jni或C++的问题，而且经我使用，函数本身的bug是很少的，出问题几乎总是内存崩溃，解决的办法也就很简单了，分段给。

jni内部的全局变量，用数组会出问题，重复调用它会死机，但基本数据类型不怕这个。

用到jni的Android程序还有个怪癖，不让改目录。你改了以后，它会失去项目结构，打开后就无法显示目录树也无法调试，即使你恢复了原来的文件夹名，它也无法恢复了。在做jni之前，就要想好文件目录，不变了。时时备份也是必须的，google自己都说，NDK是个实验性的东西，我们就更不知道它会出什么妖蛾子。

grayhat

发表于 2016-1-15 00:18 | 来自 [51CTO网页](#)

[只看他] 12#

【发布】

像普通的Android项目一样，“Build > Generate Signed APK”，按提示进行。

新新人类

- 帖子 13
- 精华 0
- 无忧币 52
- 个人空间 发消息
- 家园好友 他的博客
- 他的资源
- 他的课程中心

结果有什么不同呢？刚才那些C++代码体现在哪里呢？你找到发布的apk文件，用WinRAR打开，里面的lib文件夹有一堆子文件夹，每个里面都有一个后缀为so的文件，它就是C++的输出结果。之所以有这么多so，是因为手机CPU有多种，一部手机打开程序时会自动寻找与它匹配的so库来加载。

即使没有发布，在调试后，项目的“app\build\outputs\apk”目录下也会有调试版的apk生成，里面同样有那些装着so文件的文件夹。

因此，so文件是裹在apk里面发给用户的，并不像Windows的dll那样放在可执行文件外面。

对于so文件的调用，Android做得比较周到，该链接的都在java和C++代码里接好了，不会出现Windows经常出现的那种动态链接库又依赖其他库而客户的机器上没有这些库而无法打开的情况。

但是有一点不方便，Android的so库不能单独制作，至少目前官网上说还不支持单独生成so库，它必须在Android项目里与java代码一起编译，这样就很慢，而且要改一点点东西也要把整个包含java和C++的项目都打开。

有一种叫“VisualGDB”的软件，可以作为VisualStudio的插件来编写带C++的Android项目，写的过程要快些了，但输出一样慢，因为它还是在用Android的SDK。而且它编代码的自由度比AndroidStudio差很远。

wangluoziyuan

发表于 2016-2-15 17:47 | 来自 51CTO网页

[只看他] 13#

👍，详细

- 新新人类
- 帖子 1
- 精华 0
- 无忧币 14
- 个人空间 发消息
- 家园好友 他的博客
- 他的资源
- 他的课程中心

« 上一贴：黑马Android52期 黑马Android46期 35期+黑马Android2 ... | 下一贴：我是互联网公司运营的，公司APP的一个体验问题。 ... »

◀ 返回列表

➔ 高级回复

+ 新 帖 ▾

快速回复主题

您需要登录后才可以回帖 [登录](#) | [立即注册](#)

神奇的代码注释



[关于我们](#) | [诚聘英才](#) | [联系我们](#) | [网站大事](#) | [友情链接](#) | [意见反馈](#) | [网站地图](#)

Copyright©2005-2014 51CTO.COM

本论坛言论纯属发布者个人观点，不代表51CTO网站立场！如有疑义，请与管理员联系：bbs@51cto.com

JD. 京东
.COM

春添尚新 新衣低至7折 随买随穿



站长统计

