第一章：基础理论知识学习

交叉编译。

JNI接口的学习。

NDK:允许使用原生代码语言（C或C++）,允许使用C++实现应用程序部分功能。

Public class MyActivity extends

官方帮助文档：

<https://developer.android.google.cn/studio/projects/add-native-code.html>

平台之间移植应用等等六点好处

具体用途比如一些算法库，游戏中的三角形，四边形的碰撞判定。

Java 虚拟机就是 C 、C++ 实现的，不支持两个之间调用，也说不过去。类似与我用 C 、 C++ 语言实现了一套解析固定格式的文件，然后调用对应的本地方法运行。

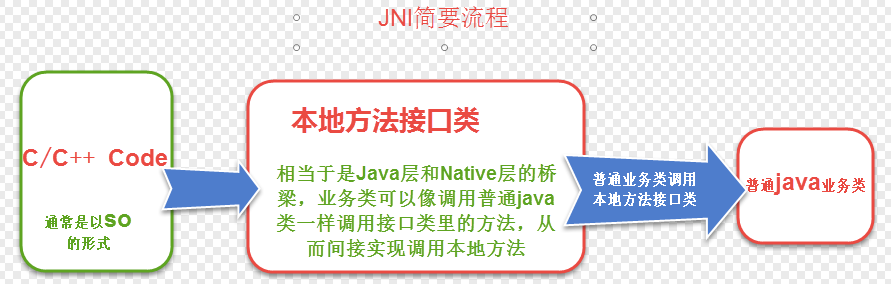
OpenGL ES2.0 、 Cocos2d-x 、FFmpeg 等开发过程中，需要使用 C 、C++ ，而 Android 本身语言是 Java 。 比如 FFmpeg 移植过来，有时会使用 SDL 框架进行渲染，而这个是 C 语言编写的。

Linux下能执行的函数库So文件，本质就是一堆C、C++的头文件和实现文件打包成一个库。

通过JNI实现Java代码与C/C++代码的交互

第二张：一个简单的栗子介绍了JNI（电子书）。

（针对不熟悉JNI初学者教程）



**JNI简要流程**

JNIExport jstring JNICALL Java\_com\_example\_hellojni\_MainActivity\_stringFromJNI( JNIEnv\* env,jobject thiz )

jstring 是**返回值类型**  
Java\_com\_example\_hellojni 是**包名**  
MainActivity 是**类名**  
stringFromJNI 是**方法名**

JNIExport和JNICALL是关键字，不修改

加载jni动态库时，同时调用jni库的一个函数JNI\_onLoad，

以一个计算器为例进行实验

注意事项

不能有module.java这个文件，或者JavaSE版本不能太低

在Eclipse点击RUN就能生成class文件，在生成.h文件

生成class文件.javac \*\*\*\*.java(程序中含有注释的话，会产生乱码)。

/\*\*

\*

\*注释内容、\*

\*

/

这样可以避免注释出现乱码，无法编译成class文件的问题。

javah -classpath . -jni 包名.类名(com.test.jni.Java2cpp),然后生成了com\_test\_jni\_Java2cpp.h

javah -classpath . -jni javajni.javajni

JAVASE-1.7成功生成.h文件，后面需要再次测试能支持的最高版本是多少

用VS2019生成dll文件，函数用C++写，再在JNI中重新复写一遍。其中要选择“具有导出项的dll链接库”（VS2019不可行，2017才行）。

在VS2019中导入eclipse生成.h文件时，发现其中的代码“#include <jni.h>”会报错”无法打开源文件“

解决：添加这两条附加目录,JDK/include/ JDK/include/win32

CTestJNI类的构造函数被我注释掉了，不知道有什么用，生成了一个dll动态链接库了。试图在eclipse里面调用这个dll，加载失败，显示 a java exception has occurred（复制到bin同级目录下面可以使用）

发生了函数报错的问题

如果C++的dll十分复杂，还依赖其他库，提示”can't find dependent libraries“把所有一来库放到同一级目录下，然后当前目录设置为库目录。

VS推荐包含pch.h头文件

进入bin界面的方法

D: 然后一直cd User ....等等执行java -h（包名.类名）

是不是eclipse生成的.h文件有问题呢，值得怀疑一下。貌似是.h文件写错了，因该有其他内容的

头文件应该是没错的，在jni中重写函数写错了头文件出错了，里面应该是有函数的，之前的没有函数

看看是不是有外部依赖项的原因；修改运行库原来是DLL(/MDd)模式。也不是这个原因

成功运行了，没有加package，是不是这个原因呢，有带考证

变化点：没加package，事实证明是不能加package的，只能用默认的package。原因未知，然后就是一定是放在最外面一层文件夹里面的，

.h文件在哪里生成都可以的，但是一定要先运行一下，不然.h文件里面会没有要用的naive函数，这是值得注意的一点。

javac -h . hello.java（对于JDK版本高于10的版本，应该使用该命令，因为javah命令已经被移除。）

对于没有包名的文件：javah 文件名（例：javah HelloJNI）

JNA:

缺点：缺少移植性，操作较为复杂。

JNI接口太多，将近2000行，把jstring变为string就得花很大力气。

在java中访问一个没有包名的函数只能通过复制该java文件到当前要调用的文件夹之中。（因为加了包名的话，就莫名其妙的不能使用jni）一个包名之中只能有一个class文件。

加载dll路径：绝对路径和相对路径

问题：有闪退

问题：怎么能确定加在了JNI这个dll文件呢（网上说要放在lib文件夹中）

在java文件下建立.h文件， javah -jni com.hello\_world.hello\_jni

先把jni学会，再去学习使用jna

问题；找不到.so文件在哪里，或者根本就没有生成。

App下的build.gradle下进行了部分修改操作。

Ndk节点：

ndk**{** moduleName "JNI"  
 abiFilters "armeabi","armeabi-v7a","x86"  
**}**

在gradle.propertity下添加了代码

android.useDeprecatedNdk=true

生成.so文件：51CTO学院：通过命令行文进入含有.mk和.cpp的文件夹，然后通过含有ndk-build的路径之内进行ndk-build。

**首先**检查local.properties文件中是否有NDK路径，如果有没有NDK路径，则添加NDK路径，比如我的如下：

解决思路：直接修改后缀名把txt文件改成.mk文件（通过实验，可以直接修改恒.mk文件，然后需要加Android.mk和Application.mk来进行生成.so）

mk文件内容解释：库名（涉及到后面的调用）， （.c文件）用来生成.so

要加上一句android-16不然ndk会报错

安卓是不是不能使用dll文件。（可以使用，但应该是个很鸡肋的技术）

Java可以识别dll，但安卓平台是不能实现的（除非用ndk工具包）

NDK是用来快速开发.so的

通过ndk-build产生了.so文件，并包含在lib之中，但是带来的问题就是要修改加载lib的定义这里，有位博主写的比较详细

<https://blog.csdn.net/FZeroTHero/article/details/89286139>

（与app同级目录的话）：一：如果在app文件夹内无法创立jni文件夹，那直接去explore里面创建。

二：在app的build.gradle文件中的android节点添加内容：

**为了指定我们要去加载的库位置。**和默认的编译在app下的build.gradle下进行。

将.so文件加载到APK之中去。

sourceSets **{** main **{** jniLibs.srcDirs = ['libs']//指定使用libs下的jni  
 jni.srcDirs = []  
 **}  
}**

配置external tools可以不用再用命令行来产生.so文件了，以后有机会再学

添加ndk路径(不知道为什么没有检索到)

1.**在**Gradle**的** local.properties**中添加配置**

ndk.dir=D\:\\SDK\\ndk-bundle

1. 添加对旧版本的支持（Gradle**的** gradle.properties**中添加配置**）

android.useDeprecatedNdk=true // 对旧版本的NDK支持

1. 两种说法（app下的**build.gradle还是gradle下的build.gradle**）应该是app下的

添加以下代码

*main* **{** jniLibs.srcDirs = ['libs']  
 jni.srcDirs = []

SDK Manager中找SDK Tools，在android里面调成project项目，可以详细看各文件夹细节。

名称不要写lib前缀和.so后缀

.mk文件，第一行是去当前目录（一般是jni目录）里找C的源代码

第二行是清除之前的.so库文件

第三行是生成MODULE so文件的名字

第四行是名字（c或c++代码的名字）

第五行生成.so的库文件。或者生成.a的库文件。

c代码写错：生成的.so文件应用在app会出现闪退的情况。

默认都使用UTF-8代码

一旦修改c代码，必须重新编译.so文件。

JDK1.7、1.8在src目录下执行javah 包名.类名

本地支持的快速用法（add native support）

1. 添加ndk路径
2. 项目右键，添加本地支持。
3. 自动生成jni文件夹

提示：ndk-build clean清除.so文件。

提示：jobject和jclass的区别就是（加上static关键字）

AS卡死异常：

AS安装目录的bin目录下，找到**idea.properties最后一行**添加

disable.android.first.run=true

问题：台式电脑无法使用gradle。于是使用了as推荐的jdk路径（但路径并没有改变）。

#### 引入第三方so库

将so库和头文件拷贝到对应目录

修改CMakeLists.txt文件

忘记在app下的build 内的source等等，导致app闪退。

不应该讲API等级升到R，升到10最为方便。

ndk.dir=D\:\\Qt\_andriod\_NDK\\android-ndk-r21b

讲清楚ndk的位置

AndroidManifest.xml在哪里？

找不到source文件，CMake无法按照路径来找到native-lib。

.so在imeta之内。

set( OpenCV\_DIR D:\\OpenCV-android-sdk\\sdk\\native\\jni )

相比较之下，多了一句设置路径的话。（让CMake找module\_path）配置opencv环境。将路径前面的话删除之后可以正常通过编译，但是会崩溃。删除了

src/main/cpp

这个路径，竟然可以跑，但崩溃了。再调试！

使用load绝对路径来加载native-lib这个.so文件。失败。

尝试复制到该目录下来加载loadlibrary

加（加载opencv\_java4库）

在CMakelist里面添加ndk的路径，还是app崩溃

ndk**{** abiFilters 'x86','x86\_64','armeabi-v7a','arm64-v8a'  
**}**

问题应该找到了：新建的一个.java文件，用完native函数之后，什么都没有返回，因此崩溃。

Java不能通过类 对象来定一个对象，不然无法调用。

只能new一个，app才会不崩溃。

Java中定义变量没有实际空间，new一个才会开辟实际空间

遇到的坑就是三个：1，camke中不需要指定native-cpp的路径

2，变量定义方法 3，头文件的生成无法把native函数单独摘出来。