例子试验可以运行

 Linux下 JNI的使用

       学习Android其中涉及对JNI的使用，对于这种跨语言的调用真没有见过，

Java也都是最近才学的更别说对JNI的了解了，

JNI的使用对于Android来说又是十分的重要和关键。那么到底Java到底是如何调用C/C++的，

通过网络达人的总结中学习，自己也顺便总结一下这个学习的过程。

**什么是JNI**

     JNI是Java native interface的简写，可以译作Java原生接口。

Java可以通过JNI调用C/C++的库，这对于那些对性能要求比较高的Java程序无疑是一个福音。

JNI是Java与C/C++交互的接口。

      使用JNI也是有代价。大家都知道JAVA程序是运行在JVM之上的，可以做到平台无关。

但是如果Java程序通过JNI调用了原生的代码（比如c/c++等），则Java程序就丧失了平台无关性。

最起码需要重新编译原生代码部分。所以应用JNI需要好好权衡，不到万不得已，请不要选择JNI，

可以选择替代方案，比如TCP/IP进行进程间通讯等等。这也是为什么谷歌的Android平台的底层虽然用JNI实现，

但是他不建议开发人员用JNI来开发Android上面的应用的原因。将会丧失Android上面的应用程序平台无关性。

代码实例

      下面以HelloWorld的实现学习Linux下 JNI的使用。

**第一步：**

创建一个 TestJni.java文件

[复制代码](javascript:void(0);)

import java.util.\*;

public class TestJni

{

//声明原生函数：参数为String类型

public native void print(String content);

//加载本地库代码

static

{

System.loadLibrary("TestJni");

}

}

[复制代码](javascript:void(0);)

编译 TestJni.java文件：javac TestJni.java

在当前文件夹下生成TestJni.class文件

      注意print方法的声明，关键字native表明该方法是一个原生代码实现的。

另外注意static代码段的System.loadLibrary调用，这段代码表示在程序加载的时候，自动加载libTestJni.so库。

**第二步：**

生成 TestJni.h文件

**执行命令：javah -jni TestJni**

      生成TestJni.h文件

[复制代码](javascript:void(0);)

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

#include <jni.h>

/\* Header for class TestJni \*/

#ifndef \_Included\_TestJni

#define \_Included\_TestJni

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: TestJni

\* Method: print

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

\*/

JNIEXPORT void JNICALL Java\_TestJni\_print

(JNIEnv \*, jobject, jstring);

#ifdef \_\_cplusplus

}

#endif

#endif

[复制代码](javascript:void(0);)

该文件中包含了一个函数Java\_TestJni\_print的声明。这里面自动包含两个参数，非常重要。JNIEnv \*和 jobject

**第三步：**

      创建TestJni.c文件

[复制代码](javascript:void(0);)

#include <jni.h>

#include <stdio.h>

#include <TestJni.h>

JNIEXPORT void JNICALL

Java\_TestJni\_print(JNIEnv \*env,jobject obj, jstring content)

{

// 从 instring 字符串取得指向字符串 UTF 编码的指针

//注意C语言必须(\*env)-> C++ env->

const jbyte \*str =

(const jbyte \*)(\*env)->GetStringUTFChars(env,content, JNI\_FALSE);

　　printf("Hello---->%s\n",str);

// 通知虚拟机本地代码不再需要通过 str 访问 Java 字符串。

(\*env)->ReleaseStringUTFChars(env, content, (const char \*)str );

return;

}

[复制代码](javascript:void(0);)

//这里看到  **JNIEnv**作用了，使得我们可以使用Java的方法

//jobject 指向在此 Java 代码中实例化的 Java 对象 LocalFunction 的一个句柄，相当于 this 指针

参数类型 jstring 对应java中的String，这里是有所不同的。每一个Java里的类型这里有对应的与之匹配。

**命令行输入：**

**cc -I/usr/lib/jvm/java-6-sun/include/linux/**

**-I/usr/lib/jvm/java-6-sun/include/**

**-I/home/xmp/AndroidProject/apk/JNI**

**-fPIC -shared -o libTestJni.so TestJni.c**

生成：libTestJni.so库文件

　　在当前目录生成libTestJni.so。注意一定需要包含Java的include目录（请根据自己系统环境设定），

因为libTestJni.c中包含了jni.h。另外一个值得注意的是在libTestJni.java中我们LoadLibrary方法加载的是“TestJni”，

可我们生成的Library却是libTestJni。这是Linux的链接规定的，

一个库的必须要是：lib+库名+.so。链接的时候只需要提供库名就可以了

　　-I/home/xmp/AndroidProject/apk/JNI 是我自己的练习目录也必须包含，否则.c文件中会找不到TestJni.h头文件。

　　现在 liblibTestJni.so就是一个可以使用的库了，其功能就是有一个print函数 与刚才所创建的TestJni.java文件对应，

TestJni.java类中加载库liblibTestJni.so，声明了其函数print。所以现在TestJni.java中具备使用print函数的功能；

所以现在我们就可以通过使用TestJni.java来使用调用C库libTestJni.so中的函数

当然现在任何java类都可已加载liblibTestJni.so库来使用其中的功能。

**第四步：**

     创建HelloWord.java函数

[复制代码](javascript:void(0);)

import java.util.\*;

public class HelloWorld

{

public static void main(String argv[])

{

new HelloWorld();

}

public HelloWorld()

{

new TestJni().print("Hello,World !"); //调用TestJni的原生函数print

}

}

[复制代码](javascript:void(0);)

**输入命令编译： javac HelloWorld.java**

生成HelloWorld.class

**第五步：**

      运行HelloWorld程序

**命令行输入：java HelloWorld**

输出结果：Hello---->Hello,World !

验证OK！

如果你这步发生问题，如果这步你收到java.lang.UnsatisfiedLinkError异常，可以通过如下方式指明共享库的路径：

**java -Djava.library.path='.' HelloWorld**

或者输入命令：

**export LD\_LIBRARY\_PATH=“HelloWorld路径”:$LD\_LIBRARY\_PATH   设置**环境变量

然后再 java HelloWorld 一样OK

简单例子，照着以下参考文档即可实现。

参考文档：

<http://my.unix-center.net/~Simon_fu/?p=359>

<http://www.ibm.com/developerworks/cn/java/l-linux-jni/>

JNIEnv功能参考：

<http://docs.oracle.com/javase/1.5.0/docs/guide/jni/spec/functions.html>