<http://www.cnblogs.com/moon1992/p/5260226.html>

**阅读目录**

* [一. 什么是JNI](http://www.cnblogs.com/moon1992/p/5260226.html#_label0)
* [二. 编写带有native声明的方法的java类](http://www.cnblogs.com/moon1992/p/5260226.html#_label1)
* [三. 使用javac命令编译所有的java类](http://www.cnblogs.com/moon1992/p/5260226.html#_label2)
* [四. 使用javah命令生成.h文件](http://www.cnblogs.com/moon1992/p/5260226.html#_label3)
* [五. 使用C/C++实现本地方法](http://www.cnblogs.com/moon1992/p/5260226.html#_label4)
* [六. 将C/C++编写的文件生成动态链接库](http://www.cnblogs.com/moon1992/p/5260226.html#_label5)

　　在上一篇文章中介绍了Java语言中的native关键字，以及Java语言调用C语言的编译生成本地动态链接库(DLL)实现加法运算的小例子，本文通过一个更加详细的例子，深入讲解Java语言调用C语言的函数。

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 一. 什么是JNI

[JNI](http://baike.baidu.com/link?url=hQbBBKp6L3CbkySA6Q69KkbkoMzZhBDYV3xDYfVIM8qqWzwcqczZBcgaXewFlVQ5CJVJuAUiARTIcjZCDZ3vS_)是Java Native Interface的缩写，它提供了若干的API实现了Java和其他语言的通信（主要是C&C++）。从Java1.1开始，JNI标准成为java平台的一部分，它允许Java代码和其他语言写的代码进行交互。JNI一开始是为了本地已编译语言，尤其是C和C++而设计的，但是它并不妨碍你使用其他编程语言，只要调用约定受支持就可以了。使用java与本地已编译的代码交互，通常会丧失平台可移植性。但是，有些情况下这样做是可以接受的，甚至是必须的。例如，使用一些旧的库，与硬件、操作系统进行交互，或者为了提高程序的性能。JNI标准至少要保证本地代码能工作在任何Java 虚拟机环境下。

　　总的来说，JNI就是一个允许Java语言和其他编程语言(主要是C/C++)通信的接口。C/C++是系统级的编程语言, 可以用来开发任何和系统相关的程序和类库, 但是Java本身编写底层的应用比较难实现, 使用JNI可以调用现有的本地库, 极大地灵活了Java的开发. C/C++的效率是目前最好的语言, 可以使用C/C++来实现一些实时性非常高的部分. C/C++和Java本身都是非常流行的编程语言, 一些大型软件中经常使用语言之间的混合编程.

　　一旦使用JNI, JAVA程序就丧失了JAVA平台的两个优点: 程序不在跨平台。要想跨平台，必须在不同的系统环境中重新编译本地语言部分；程序不再是绝对安全的，本地代码的不当使用可能导致整个陈旭崩溃。一个通用的规则是，你应该让本地方法集中在少数几个类当中，这样就降低了Java语言和C/C++之间的耦合性。

　　使用JNI实现Java与C语言混合编程的基本步骤如下：

1. 编写带有native声明的方法的java类
2. 使用javac命令编译所有的java类
3. 然后使用javah + 类名生成扩展名为.h的头文件
4. 使用C/C++实现本地方法
5. 将C/C++编写的文件生成动态链接库

　　下面就完整的介绍一个简单的Java调用C/C++的例子, 这个例子参考了[[点击这里跳转原始例子](http://www.cnblogs.com/icejoywoo/archive/2012/02/22/2363709.html)], 并对这个例子做了修改。

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 二. 编写带有native声明的方法的java类

　　首先编写一个文件名为Sample的java程序，代码如下:

[复制代码](javascript:void(0);)

1 public class Sample

2 {

3 public static void main(String[] args)

4 {

5 System.loadLibrary("SampleC");

6 NativeMethodTest nmt = new NativeMethodTest();

7

8 int square = nmt.intMethod(5);

9 boolean bool = nmt.booleanMethod(true);

10 String text = nmt.stringMethod("java");

11 int sum = nmt.intArrayMethod(new int[]{1,2,3,4,5,6,7,8,13});

12

13 System.out.println("intMethod: " + square);

14 System.out.println("booleanMethod:" + bool);

15 System.out.println("stringMethod:" +text);

16 System.out.println("intArrayMethod:" + sum);

17 }

18 }

19

20 class NativeMethodTest

21 {

22 public native int intMethod(int n);

23 public native boolean booleanMethod(boolean bool);

24 public native String stringMethod(String text);

25 public native int intArrayMethod(int[] intArray);

26 }

[复制代码](javascript:void(0);)

　　上面NativeMethodTest类中有四个native关键字声明的方法，分别是4种类型的参数, int, boolean, String, int[]. 其中 System.loadLibrary("SampleC"); 这行代码加载了本地动态链接库，在Windows系统下加载的就是SampleC.dll, 在Linux系统下加载的就是SampleC.so。注意不可以在代码中写上扩展名.dll或者.so。本文使用的是Windows系统，加载的是SampleC.dll，但是这个DLL文件就是我们稍后要使用C/C++编译生成的文件。

　　这4个native关键字修饰的方法就是需要我们用C/C++实现的方法。

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 三. 使用javac命令编译所有的java类

　　编译Sample.java，使用命令行(Windows下是cmd) javac Sample.java 可以看到生成两个class文件：Sample.class, NativeMethodTest.class

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 四. 使用javah命令生成.h文件

　　在命令行中运行 javah NativeMethodTest 可以生成NativeMethodTest.h文件，注意不是 javah Sample , native关键字修饰的方法在那个类中就使用javah命令生成对应的头文件。NativeMethodTest.h的代码如下：

[复制代码](javascript:void(0);)

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

2 #include <jni.h>

3 /\* Header for class NativeMethodTest \*/

4

5 #ifndef \_Included\_NativeMethodTest

6 #define \_Included\_NativeMethodTest

7 #ifdef \_\_cplusplus

8 extern "C" {

9 #endif

10 /\*

11 \* Class: NativeMethodTest

12 \* Method: intMethod

13 \* Signature: (I)I

14 \*/

15 JNIEXPORT jint JNICALL Java\_NativeMethodTest\_intMethod

16 (JNIEnv \*, jobject, jint);

17

18 /\*

19 \* Class: NativeMethodTest

20 \* Method: booleanMethod

21 \* Signature: (Z)Z

22 \*/

23 JNIEXPORT jboolean JNICALL Java\_NativeMethodTest\_booleanMethod

24 (JNIEnv \*, jobject, jboolean);

25

26 /\*

27 \* Class: NativeMethodTest

28 \* Method: stringMethod

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

30 \*/

31 JNIEXPORT jstring JNICALL Java\_NativeMethodTest\_stringMethod

32 (JNIEnv \*, jobject, jstring);

33

34 /\*

35 \* Class: NativeMethodTest

36 \* Method: intArrayMethod

37 \* Signature: ([I)I

38 \*/

39 JNIEXPORT jint JNICALL Java\_NativeMethodTest\_intArrayMethod

40 (JNIEnv \*, jobject, jintArray);

41

42 #ifdef \_\_cplusplus

43 }

44 #endif

45 #endif

[复制代码](javascript:void(0);)

 　　这个h文件相当于我们在java里面的接口，这里声明了四个方法，然后在我们的本地方法里面实现这个方法，也就是说我们在编写C/C++程序的时候所使用的方法名必须和这里的一致。

　　在注释中我们可以看到这样一个东西 Signature, 这个是方法的签名. 关于Signature, 下面通过一个表格来说明.

|  |  |  |
| --- | --- | --- |
| java类型 | Signature | 备注 |
| boolean | Z |  |
| byte | B |  |
| char | C |  |
| short | S |  |
| int | I |  |
| long | L |  |
| float | F |  |
| double | D |  |
| void | V |  |
| Object | L用/分割的完整类名 | 例如: Ljava/lang/String表示String类型 |
| Array | [签名 | 例如: [I表示int类型的数组, [Ljava/lang/String表示String类型的数组 |
| Method | (参数签名)返回类型签名 | 例如: ([I)I表示参数类型为int数组，返回值int类型的方法 |

　　在每个函数的参数列表中都有JNIEnv \*和 jobject两个参数, 这两个参数稍候说明.

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 五. 使用C/C++实现本地方法

　　可以使用C语言来实现, 也可以使用C++来实现, 下面说说C语言的实现.

　　创建DLL工程，添加NativeMethodTest.h等头文件，具体方法在这一篇随笔中有详细介绍[[点击查看](http://www.cnblogs.com/moon1992/p/5258245.html)]，C程序如下：

[复制代码](javascript:void(0);)

1 # include "NativeMethodTest.h"

2 # include <string.h>

3

4 JNIEXPORT jint JNICALL Java\_NativeMethodTest\_intMethod

5 (JNIEnv \* env, jobject obj, jint i)

6 {

7 return i \* i;

8 }

9

10 JNIEXPORT jboolean JNICALL Java\_NativeMethodTest\_booleanMethod

11 (JNIEnv \* env, jobject obj, jboolean b)

12 {

13 return !b;

14 }

15

16 JNIEXPORT jstring JNICALL Java\_NativeMethodTest\_stringMethod

17 (JNIEnv \* env, jobject obj, jstring string)

18 {

19 const char\* str = (\*env)->GetStringUTFChars(env, string, 0);

20 char cap[128];

21 strcpy(cap, str);

22 (\*env)->ReleaseStringUTFChars(env, string, 0);

23 return (\*env)->NewStringUTF(env, strupr(cap));

24 }

25

26 JNIEXPORT jint JNICALL Java\_NativeMethodTest\_intArrayMethod

27 (JNIEnv \* env, jobject obj, jintArray array)

28 {

29 int i, sum = 0;

30 jsize len = (\*env)->GetArrayLength(env, array);

31 jint \* body = (\*env)->GetIntArrayElements(env, array, 0);

32

33 for (i=0; i<len; i++)

34 {

35 sum += body[i];

36 }

37 (\*env)->ReleaseIntArrayElements(env, array, body, 0);

38 return sum;

39 }

[复制代码](javascript:void(0);)

　　(\*env)->GetStringUTFChars()这个方法, 是用来在Java和C之间转换字符串的, 因为Java本身都使用了双字节的字符, 而C语言本身都是单字节的字符, 所以需要进行转换.

　　JNIEnv \*是每个函数都有的参数, 它包含了很多有用的方法, 使用起来类似Java的反射, 也提供了这样一个编码转换的函数.

　　GetStringUTFChars()和NewStringUTF(), 第一个是从UTF-8转换为C的编码格式, 第二个是根据C的字符串返回一个UTF8字符串.

　　ReleaseStringUTFChars()是用来释放对象的, 在Java中有虚拟机进行垃圾回收, 但是在C语言中, 这些对象必须手动回收. 否则可能造成内存泄漏.

　　函数的名字一眼看到就可以猜出功能, jni.h中的大部分函数名都是这样.

[回到顶部](http://www.cnblogs.com/moon1992/p/5260226.html#_labelTop)

### 六. 将C/C++编写的文件生成动态链接库

　　将编译生成的SampleC.dll文件拷贝到Sample.class所在的文件夹中，在命令行中运行：java Sample 即可看到如下结果：

intMethod: 25

booleanMethod:false

stringMethod:JAVA

intArrayMethod:49

参考文献：

http://www.cnblogs.com/icejoywoo/archive/2012/02/22/2363709.html

http://www.cnblogs.com/moon1992/

分类: [Java](http://www.cnblogs.com/moon1992/category/692025.html)