Linux 下 JNI 调用简单实例操作全过程

snowdream <yanghui1986527@gmail.com>

开发环境：Linux(Ubuntu 11.04) + JDK 7

实例说明：利用 JNI 调用本地代码的方法来实现一个计算 Int 数组总和的功能

使用 JNI 调用本地代码，整个开发流程主要包括以下几个步骤：

1、创建一个 Java 类（IntArray.java）；

2、使用 javac 编译该类（生成 IntArray.class）；

3、使用 javah -jni 产生头文件（生成 IntArray.h）;

4、使用本地代码实现头文件中定义的方法（编写 IntArray.c）;

5、编译生成本地动态库（生成 libIntArray.so）；

6、使用 Java 运行程序。

一、创建一个 Java 类（IntArray.java）

```
class IntArray{
	private native int sumArray(int[] arr);
	public static void main(String[]args){
		IntArray p = new IntArray();
		int arr[] = new int[10];
		for(int i =0;i<10;i++){
			arr[i] = i;
		}
		int sum = p.sumArray(arr);
		System.out.println("Sum = "+sum);
	}

	static{
		System.loadLibrary("IntArray");
		}

}
```

注：

1、在 Java 代码中声明本地方法必须有"native"标识符，native 修饰的方法，在 Java 代码中只作为声明存在。例如： private native int sumArray(int[] arr);

2、在调用本地方法前，必须首先装载含有该方法的本地库. 如 IntArray.java 中所示，置于 static 块中，在 Java VM 初始化一个类时，首先执行这部分代码，这可保证调用本地方法前，装载了本地库。

```
static{
        System.loadLibrary("IntArray");
        }
```

二、使用 javac 编译该类（生成 IntArray.class）

```
javac IntArray.java
```

三、使用 javah -jni 产生头文件（生成 IntArray.h）

```
javah -jni IntArray
```

生成 IntArray.h

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

#ifndef _Included_IntArray
#define _Included_IntArray
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     IntArray
 * Method:    sumArray
 * Signature: ([I)I
 */
```

```
JNIEXPORT jint JNICALL Java_IntArray_sumArray
  (JNIEnv *, jobject, jintArray);
#ifdef __cplusplus
}
#endif
#endif
```

四、使用本地代码实现头文件中定义的方法（编写 IntArray.c）

复制 IntArray.h 成 IntArray.c,对于 IntArray.c 做以下修改：

1、添加头文件：#include "IntArray.h"

2、去掉以下几句：

```
#ifndef _Included_IntArray
#define _Included_IntArray
#endif
```

3、实现头文件中定义的方法

IntArray.c 具体代码如下：

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

#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     IntArray
 * Method:    sumArray
 * Signature: ([I)I
 */
JNIEXPORT jint JNICALL Java_IntArray_sumArray
  (JNIEnv *env, jobject obj, jintArray arr)
```

```
{
        jint buf[10] ={0};
        jint i = 0,sum = 0;

        (*env)->GetIntArrayRegion(env,arr,0,10,buf);

        for(i=0;i<10;i++)
        {
                sum += buf[i];
        }

        return sum;
}
#ifdef __cplusplus
}
#endif
```

五、编译生成本地动态库（生成 libIntArray.so）

根据本地代码(IntArray.h,IntArray.c)生成本地动态库，命令如下：

```
gcc -I/usr/lib/jvm/java-7-sun/include/ -I/usr/lib/jvm/java-7-sun/include/linux/ -fPIC -shared -o libIntArray.so IntArray.c
```

注：

其中 -I 后面是 java 的 include 文件夹地址，请根据您具体的 java 版本以及安装路径作相应改变；

-f 后面的 PIC 表示生成的库中符号是与位置无关的(PIC 就是 Position Independent Code)，关于 PIC，可以参考这篇文章

Introduction to Position Independent Code

-shared 表示共享，共享库后缀名.so 可以认为是 shared object 的简称；

-o libIntArray.so，可以理解为编译后输出 libIntArray.so 库。

六、使用 Java 运行程序

```
java IntArray
```

可能出现以下结果：

```
snowdream@snowdream:~$ java IntArray
Exception in thread "main" java.lang.UnsatisfiedLinkError: no IntArray in java.library.path
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1860)
        at java.lang.Runtime.loadLibrary0(Runtime.java:845)
        at java.lang.System.loadLibrary(System.java:1084)
        at IntArray.<clinit>(IntArray.java:15)
```

分析异常提示可知，我们之前生成的共享库不在系统默认的共享库路径中，程序找不到共享库报错。

解决方法有两个：

1、临时指定共享库 libIntArray.so 的路径。

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

注：该方法只在当前会话窗口有效，切换到另外一个终端窗口，则需要重新指定共享库路径。

2、运行时加上参数指定共享库 libIntArray.so 的路径。

```
java -Djava.library.path=.  IntArray
```

注：-D：设置 Java 平台的系统属性。 此时 JavaVM 可以在当前位置找到该库。

通过以上任意方法，您都可以得到正确的运行结果：

```
snowdream@snowdream:~$ java IntArray
Sum = 45
```