JNI 技 术 之 数 据 类 型

作者:程姚根 联系方式:chengyaogen123@163.com

前面两节我们看了一下如何将JAVA 层的"native"函数与JNI层的函数关联起来，我们介绍了JNI技术提供的动态注册机制和静态注册机制。最终我们达到的效果是，JAVA层可以调用JNI层的函数。

细心的读者肯定会问，JAVA层如果传递参数到JNI层，那如何是好，要知道JAVA的数据类型和C/C++的数据类型还是有一些区别的。那在JNI层如何表示JAVA层传递下来的参数呢?下面我就一起来看一下JNI技术对JAVA层传递的参数进行处理的机制。

JNI层数据类型和JAVA层数据类型的对应关系

(1) 基本类型对应关系

Java Language Type	Native Type	Description
boolean	jboolean	unsigned 8 bits
byte	jbyte	signed 8 bits
char	jchar	unsigned 16 bits
short	jshort	signed 16 bits
int	jint	signed 32 bits
long	jlong	signed 64 bits
float	jfloat	32 bits
double	jdouble	64 bits

打开jni.h和jni_md.h的头文件，我们就可以看到Native Type类型是如何得到了,如下图所示:

```
 5  * SUN PROPRIETARY/CONFIDENTIAL. Use
 6  */
 7
 8 #ifndef _JAVASOFT_JNI_MD_H_
 9 #define _JAVASOFT_JNI_MD_H_
10
11 #define JNIEXPORT
12 #define JNIIMPORT
13 #define JNICALL
14
15 typedef int jint;
16 #ifdef _LP64 /* 64-bit Solaris */
17 typedef long jlong;
18 #else
19 typedef long long jlong;
20 #endif
21
22 typedef signed char jbyte;
23
24 #endif /* !_JAVASOFT_JNI_MD_H_ */
```

```
23
24 /* jni_md.h contains the machine-dependent typ
25    and jlong */
26
27 #include "jni_md.h"
28
29 #ifdef __cplusplus
30 extern "C" {
31 #endif
32
33 /*
34  * JNI Types
35  */
36
37 #ifndef JNI_TYPES_ALREADY_DEFINED_IN_JNI_MD_H
38
39 typedef unsigned char   jboolean;
40 typedef unsigned short  jchar;
41 typedef short       jshort;
42 typedef float       jfloat;
43 typedef double      jdouble;
```

呵呵，原来是通过typedef 把基本类型重命名了。

(2)引用类型对应关系

```
jobject                         (all objects)
    ├── jclass                  (java.lang.Class instances)
    ├── jstring                 (java.lang.String instances)
    ├── jarray                  (arrays)
    │       ├── jobjectArray        (Object[])
    │       ├── jbooleanArray       (boolean[])
    │       ├── jbyteArray          (byte[])
    │       ├── jcharArray          (char[])
    │       ├── jshortArray         (short[])
    │       ├── jintArray           (int[])
    │       ├── jlongArray          (long[])
    │       ├── jfloatArray         (float[])
    │       └── jdoubleArray        (double[])
    └── jthrowable              (java.lang.Throwable objects)
```

通过上面的图，**我们可以知道JAVA层传递给JNI层的参数如果是一个对象，那无论它在JAVA层是什么类型，到了JNI层都用jobject类型。**

在这里我们一定要注意，对于引用类型，JNI层的代码是不能直接使用的。例如jstring表示java VM中字符串，并且和普通的C字符串是不一样的，在JNI层代码中不能像使用普通C字符串一样使用它们。下面的代码，如果直接运行极可能会摧毁虚拟机。

```
JNIEXPORT jstring JNICALL
Java_Prompt_getLine(JNIEnv *env, jobject obj, jstring prompt)
{
    /* ERROR: incorrect use of jstring as a char* pointer */
    printf("%s", prompt);
    ...
}
```

既然不能直接使用，必定是需要通过一些函数完成转化才可以使用的。是的，还记得JNIEnv吗，那里面就有我们的转化函数。