JNI数据类型的详解

在Java中有两类数据类型：primitive types，如，int, float, char；另一种为reference types，如，类，实例，数组。

注意：数组，不管是对象数组还是基本类型数组，都作为reference types存在，有专门的JNI方法取数组中每个元素。

1、void

java的void与JNI的void是一致的。

2、基本数据类型

	A	B	C	D
1	基本数据类型			
2	Java类型	JNI类型	纯C/C++类型	描述
3	boolean	jboolean	unsigned char 或 uint8_t	C/C++8位整型
4	byte	jbyte	signed char 或 int8_t	C/C++有符号的8位整型
5	char	jchar	unsigned short 或 uint16_t	C/C++无符号的16位整型
6	short	jshort	short 或 int16_t	C/C++有符号的16位整型
7	int	jint	int 或 int32_t	C/C++有符号的32位整型
8	long	jlong	long long 或 int64_t	C/C++有符号的64位整型
9	float	jfloat	float @conowen	C/C++32位浮点型
10	double	jdouble	double	C/C++64位浮点型

3、对象类型

	A	B	C
1	对象类型		
2	Java类型	JNI类型	描述
3	Object	jobject	任意Java对象，或者没有对应java类型的对象
4	Class	jclass	Class对象
5	String	jstring	字符串对象 @conowen
6			

相比基本类型，对象类型的传递要复杂得多。不能对Jstring进行直接操作。
//如下使用方式是错误的，因为jstring不同于C语言中的char *类型。

```
1. Java_com_conowen_test_testActivity_test(JNIEnv *env, jobject obj, jstring s
   tr)
2. {
3. /* ERROR: incorrect use of jstring as a char* pointer */
4. printf("%s", str);
5. ...
6. }
```

注意：

typedef jint jsize;

	A	B
3	JNI String 相关函数	描述
4	const jchar * GetStringChars(JNIEnv *env, jstring str, jboolean *isCopy);	得到Unicode编码的String指针，返回值为string的copy值
5	void ReleaseStringChars(JNIEnv *env,jstring string, const jchar *chars);	释放
6	const jbyte * GetStringUTFChars(JNIEnv *env,jstring string, jboolean *isCopy);	得到UTF-8编码的String指针，返回值为string的copy值
7	void ReleaseStringUTFChars(JNIEnv *env,jstring string, const char *utf);	释放
8	jsize GetStringLength(JNIEnv *env, jstring string);	得到Unicode编码格式的String长度
9	jsize GetStringUTFLength(JNIEnv *env,jstring string);	得到UTF-8编码格式的String长度
10	jstring NewString(JNIEnv *env,const jchar *uchars, jsize len);	创建一个java.lang.String实例，长度与参数中给出的Unicode编码格式的String相同
11	jstring NewStringUTF(JNIEnv *env,const char *bytes);	创建一个java.lang.String实例，长度与参数中给出的UTF-8编码格式的String相同
12	const jchar * GetStringCritical(JNIEnv *env,jstring string, jboolean *isCopy);	得到编码格式的String的指针，返回Stirng的copy值
13	void ReleaseStringCritical(JNIEnv *env,jstring string, const jchar *carray);	释放 @conowen
14	void GetStringRegion(JNIEnv *env, jstring str,jsize start, jsize len, jchar *buf); (*env)->SetStringRegion	把String的内容复制出来，复制给Unicode编码格式的参数buf
15	void GetStringUTFRegion(JNIEnv *env, jstring str,jsize start, jsize len, jchar *buf) (*env)->SetStringUTFRegion	把String的内容复制出来，复制给UTF-8编码格式的参数buf

3.1、GetStringUTFChars与ReleaseStringUTFChars函数简单说明（跳到3.2有更方便的函数）

JNI支持Unicode/UTF-8字符编码互转。Unicode以16-bits值编码；UTF-8是一种以字节为单位变长格式的字符编码，并与7-bitsASCII码兼容。UTF-8字串与C字串一样，以NULL('\0')做结束符，当UTF-8包含非ASCII码字符时，以'\0'做结束符的规则不变。7-bit ASCII字符的取值范围在1-127之间，这些字符的值域与UTF-8中相同。当最高位被设置时，表示多字节编码。

//调用GetStringUTFChars，把一个Unicode字串转成UTF-8格式字串

```
Java_com_conowen_test_testActivity_test(JNIEnv *env, jobject obj, jstring str)
{
```

```
char buf[128];
const jbyte *cbyte;
cbyte= (*env)->GetStringUTFChars(env, str, NULL);
if (cbyte== NULL) {
return NULL;
}
printf("%s", cbyte);
(*env)->ReleaseStringUTFChars(env, str, cbyte);

scanf("%127s", buf);
return (*env)->NewStringUTF(env, buf);

//或者return (*env)->NewStringUTF(env, "hello world");
 }
```

上述函数中，有isCopy参数，当该值为JNI_TRUE，将返回str的一个拷贝；为JNI_FALSE将直接指向str的内容。 注意：当isCopy为JNI_FALSE，不要修改返回值，不然将改变java.lang.String的不可变语义。一般会把isCopy设为NULL，不关心Java VM对返回的指针是否直接指向java.lang.String的内容。

注意：在调用GetStringChars之后，一定要调用ReleaseStringChars做释放，(Unicode -> UTF-8转换的原因)。不管在调用GetStringChars时为isCopy赋值JNI_TRUE还是JNI_FALSE，因不同JavaVM实现的原因，ReleaseStringChars可能释放内存，也可能释放一个内存占用标记。

3.2、GetStringRegion/GetStringUTFRegion函数简单说明

因为这两个函数不涉及内存操作，所以较GetStringUTFChars使用要简单。也不用进行释放指针之类的操作，非常方便。（推荐使用）

```
Java_com_conowen_test_testActivity_test(JNIEnv *env, jobject obj, jstring str)
{

char outputbuf[128], inputbuf[128];
int len = (*env)->GetStringLength(env, str);
(*env)->GetStringUTFRegion(env, str, 0, len, outbuf);
printf("%s", outputbuf);
scanf("%s", inputbuf);
return (*env)->NewStringUTF(env, inbuf);
}
```

GetStringUTFRegion有两个主要的参数，start 和 length, 这两个参数以Unicode编码计算. 该函数会做边界检查，所以可能抛出StringIndexOutOfBoundsException。

3.3、GetStringLength/GetStringUTFLength函数简单说明

前者是Unicode编码长度，后者返回的是是UTF编码长度。

4、数组类型

	A	B	C
1	JNI数组类型		
2	Java类型	JNI类型	描述
3	boolean[]	jbooleanArray	布尔型数组
4	byte[]	jbyteArray	比特型数组
5	char[]	jcharArray	字符型数组
6	short[]	jshortArray	短整型数组
7	int[]	jintArray	整型数组 @conowen
8	long[]	jlongArray	长整型数组
9	float[]	jfloatArray	浮点型数组
10	double[]	jdoubleArray	双浮点型数组

JNI对每种数据类型的数组都有对应的函数。

4.1、常见错误操作：

```
/* 直接操作数组是错误的 */
Java_IntArray_sumArray(JNIEnv *env, jobject obj, jintArray arr)
{
int i, sum = 0;
for (i = 0; i < 10; i++) {
sum += arr[i];
}
}
```

4.2、使用

void Get<Type>ArrayRegion(JNIEnv *env,<ArrayType> array,

jsize start,jsize len, <NativeType> *buf);
进行操作

参数说明：

env: the JNIEnv interface pointer.

array: a reference to an array whose elements are to be copied.将要被拷贝的目标数组<ArrayType>

start: the starting index of the array elements to be copied.（数组的起始位置）

len: the number of elements to be copied.（拷贝元素的个数）buf:the destination buffer.存放结果的本地数组<NativeType>
返回值：void

Get<Type>ArrayRegion	*<ArrayType>*	*<NativeType>*
GetBooleanArrayRegion	jbooleanArray	jboolean
GetByteArrayRegion	jbyteArray	jbyte
GetCharArrayRegion	jcharArray	jchar
GetShortArrayRegion	jshortArray	jhort
GetIntArrayRegion	jintArray	jint
GetLongArrayRegion	jlongArray	jlong
GetFloatArrayRegion	jfloatArray	jloat
GetDoubleArrayRegion	jdoubleArray	jdouble

```
Java_IntArray_sumArray(JNIEnv *env, jobject obj, jintArray arr)
{
jint buf[10];
jint i, sum = 0;
(*env)->GetIntArrayRegion(env, arr, 0, 10, buf);
for (i = 0; i < 10; i++) {
sum += buf[i];
}
return sum;
}
```

JNI中数组的基类为jarray，其他如jintArray都是继承自jarray。

4.3、使用<NativeType> *Get<Type>ArrayElements(JNIEnv *env,<ArrayType> array, jboolean *isCopy);进行数组操作

Get<Type>ArrayElements	*<ArrayType>*	*<NativeType>*
GetBooleanArrayElements	jbooleanArray	jboolean
GetByteArrayElements	jbyteArray	jbyte
GetCharArrayElements	jcharArray	jchar
GetShortArrayElements	jshortArray	jshort
GetIntArrayElements	jintArray	jint
GetLongArrayElements	jlongArray	jlong
GetFloatArrayElements	jfloatArray	jfloat
GetDoubleArrayElements	jdoubleArray	jdouble

参数说明：

env: the JNIEnv interface pointer.array: a reference to the primitive array whose elements are tobe accessed.（目标数组）

isCopy: a pointer to a jboolean indicating whether a function
返回值：返回指向Java数组的一个直接的指针

使用实例：

```
Java_IntArray_sumArray(JNIEnv *env, jobject obj, jintArray arr)
{
jint *carr;
jint i, sum = 0;
carr = (*env)->GetIntArrayElements(env, arr, NULL);
if (carr == NULL) {
return 0; /* exception occurred */
}
for (i=0; i<10; i++) {
sum += carr[i];
}
(*env)->ReleaseIntArrayElements(env, arr, carr, 0);
return sum;
}
```

更多数组操作函数：

JNI Function	Description	Since
Get<Type>ArrayRegion *Set<Type>ArrayRegion*	Copies the contents of primitive arrays to or from a preallocated C buffer.	JDK1.1
Get<Type>ArrayElements *Release<Type>ArrayElements*	Obtains a pointer to the contents of a primitive array.May return a copy of the array.	JDK1.1
GetArrayLength	Returns the number of elements in the array.	JDK1.1
New<Type>Array	Creates an array with the given length.	JDK1.1
GetPrimitiveArrayCritical ReleasePrimitiveArrayCritical	Obtains or releases a pointer to the contents of a primitive array. May disable garbage collection, or return a copy of the array.	Java 2 SDK1.2

5、另外一些有用的宏定义（来自jni.h）

```
#define JNI_FALSE   0
#define JNI_TRUE    1

#define JNI_VERSION_1_1 0x00010001
#define JNI_VERSION_1_2 0x00010002
#define JNI_VERSION_1_4 0x00010004
#define JNI_VERSION_1_6 0x00010006

#define JNI_OK          (0)         /* no error */
#define JNI_ERR         (-1)        /* generic error */
#define JNI_EDETACHED   (-2)        /* thread detached from the VM */
#define JNI_EVERSION    (-3)        /* JNI version error */

#define JNI_COMMIT      1           /* copy content, do not free buffer */
#define JNI_ABORT       2           /* free buffer w/o copying back */
```