JNI 操作 java 数组

efan 关注

2018.06.11 21:33:27 字数 2,066 阅读 3,296

在 jni 函数中对 java 数组的操作主要包含以下几类：

1. GetArrayLength(jarray array)

   用于返回 java 数组的数据长度

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jintArray intArray){
    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");

    jsize size = env->GetArrayLength(intArray);
    __android_log_print(ANDROID_LOG_INFO, "native", "data size %d", size);

    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

2. Get<Type>ArrayElements(<Type>Array arr , jboolean* isCopide) 与 Release<Type>ArrayElements(<Type>Array arr , <Type>* array , jint mode)

```
jboolean* GetBooleanArrayElements(jbooleanArray array, jboolean* isCopy)
{ return functions->GetBooleanArrayElements(this, array, isCopy); }
jbyte* GetByteArrayElements(jbyteArray array, jboolean* isCopy)
{ return functions->GetByteArrayElements(this, array, isCopy); }
jchar* GetCharArrayElements(jcharArray array, jboolean* isCopy)
{ return functions->GetCharArrayElements(this, array, isCopy); }
jshort* GetShortArrayElements(jshortArray array, jboolean* isCopy)
{ return functions->GetShortArrayElements(this, array, isCopy); }
jint* GetIntArrayElements(jintArray array, jboolean* isCopy)
{ return functions->GetIntArrayElements(this, array, isCopy); }
jlong* GetLongArrayElements(jlongArray array, jboolean* isCopy)
{ return functions->GetLongArrayElements(this, array, isCopy); }
jfloat* GetFloatArrayElements(jfloatArray array, jboolean* isCopy)
{ return functions->GetFloatArrayElements(this, array, isCopy); }
jdouble* GetDoubleArrayElements(jdoubleArray array, jboolean* isCopy)
{ return functions->GetDoubleArrayElements(this, array, isCopy); }

void ReleaseBooleanArrayElements(jbooleanArray array, jboolean* elems, jint mode)
{ functions->ReleaseBooleanArrayElements(this, array, elems, mode); }
void ReleaseByteArrayElements(jbyteArray array, jbyte* elems, jint mode)
{ functions->ReleaseByteArrayElements(this, array, elems, mode); }
void ReleaseCharArrayElements(jcharArray array, jchar* elems, jint mode)
{ functions->ReleaseCharArrayElements(this, array, elems, mode); }
void ReleaseShortArrayElements(jshortArray array, jshort* elems, jint mode)
{ functions->ReleaseShortArrayElements(this, array, elems, mode); }
void ReleaseIntArrayElements(jintArray array, jint* elems, jint mode)
{ functions->ReleaseIntArrayElements(this, array, elems, mode); }
void ReleaseLongArrayElements(jlongArray array, jlong* elems, jint mode)
{ functions->ReleaseLongArrayElements(this, array, elems, mode); }
void ReleaseFloatArrayElements(jfloatArray array, jfloat* elems, jint mode)
{ functions->ReleaseFloatArrayElements(this, array, elems, mode); }
void ReleaseDoubleArrayElements(jdoubleArray array, jdouble* elems, jint mode)
{ functions->ReleaseDoubleArrayElements(this, array, elems, mode); }
```

其中 Get<Type>ArrayElements 函数可以把Java基本类型的数组转换到C/C++中的数组，有两种处理方式，一种是拷贝一份传回本地代码，另一个是把指向Java数组的指针直接传回到

本地代码中；该方法有两个参数，第一个参数指向了需要处理的数组，第二个参数用于表示把Java基本类型的数组转换到C/C++中的数组具体使用的处理方式，也就是说是否发生了 数组copy；

在完成数组数据处理后，调用 Release<Type>ArrayElements 方法，这个函数可以选择将如何处理 Java 与 C++ 的数组，是提交，还是撤销等，内存释放还是不释放等；该方法有三个参数，第一个参数指向了 java 数组，第二个参数指向调用 Get<Type>ArrayElements 函数返回的数组指针，第三个参数用于决定数组是提交，还是撤销等，内存释放还是不释放，其中包含三种模式:

1. mode = 0

- 原始数据: 对象数组将不会被限制.
- 拷贝数据: 数据将会拷贝回原始数据, 同时释放拷贝数据.

2. mode = JNI_COMMIT (1)

- 原始数据: 什么都不作.
- 拷贝数据: 数据将会拷贝回原始数据, 不释放拷贝数据.

3. mode = JNI_ABORT (2)

- 原始数据: 对象数组将不会被限制, 之前的数据操作有效
- 拷贝数据: 释放拷贝数据, 之前的任何数据操作会丢弃.

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jintArray intArray){
    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");

    jboolean isCp = JNI_FALSE;
    jint* coldata = env->GetIntArrayElements(intArray, &isCp);
    __android_log_print(ANDROID_LOG_INFO, "native", "isCopy %d", isCp);
    jsize size = env->GetArrayLength(intArray);
    for (int j = 0; j < size; j++) {
        coldata[j] += 1;
    }
    env->ReleaseIntArrayElements(intArray, coldata, 0);

    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

再具体使用过程中，先调用 GetIntArrayElements 获取 java 数组的指针以及是否发生了拷贝；然后就可以对数组数据进行处理，在上诉代码中只是做了 +1 操作，然后调用 ReleaseIntArrayElements 方法，其中 mode 为 0，也就是说将数据处理结构同步到 java 数组，并释放数组指针；

3. GetPrimitiveArrayCritical(jarray arr , jboolean* isCopied) 与 ReleasePrimitiveArrayCritical(jarray arr , void* array , jint mode)

这两个方法与上面的 Get<Type>ArrayElements 与Release<Type>ArrayElements 方法非常像，但是有一点很重要的不同：GetPrimitiveArrayCritical 更倾向于也更容易获取 java 中数组的指针而不进行拷贝，这也导致在调用 GetPrimitiveArrayCritical 与 ReleasePrimitiveArrayCritical 不能执行耗时很长的操作或者需要阻塞并等待的操作，而且在这之间 JVM 很可能会暂时禁用GC；

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jintArray intArray){

    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");
    jboolean isCp = JNI_FALSE;
    jint* coldata = static_cast<jint *>(env->GetPrimitiveArrayCritical(intArray, &isCp));
    __android_log_print(ANDROID_LOG_INFO, "native", "isCopy %d", isCp);
    jsize size = env->GetArrayLength(intArray);
    for (int j = 0; j < size; j++) {
        coldata[j] += 1;
    }
    env->ReleasePrimitiveArrayCritical(intArray, coldata, 0);

 std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

4. Get<Type>ArrayRegion(<Type>Array arr , jsize start , jsize len , <Type>* buffer)

```
void GetBooleanArrayRegion(jbooleanArray array, jsize start, jsize len, jboolean* buf)
{ functions->GetBooleanArrayRegion(this, array, start, len, buf); }
void GetByteArrayRegion(jbyteArray array, jsize start, jsize len, jbyte* buf)
{ functions->GetByteArrayRegion(this, array, start, len, buf); }
void GetCharArrayRegion(jcharArray array, jsize start, jsize len,jchar* buf)
{ functions->GetCharArrayRegion(this, array, start, len, buf); }
void GetShortArrayRegion(jshortArray array, jsize start, jsize len,jshort* buf)
{ functions->GetShortArrayRegion(this, array, start, len, buf); }
void GetIntArrayRegion(jintArray array, jsize start, jsize len,jint* buf)
{ functions->GetIntArrayRegion(this, array, start, len, buf); }
void GetLongArrayRegion(jlongArray array, jsize start, jsize len,jlong* buf)
{ functions->GetLongArrayRegion(this, array, start, len, buf); }
void GetFloatArrayRegion(jfloatArray array, jsize start, jsize len,jfloat* buf)
{ functions->GetFloatArrayRegion(this, array, start, len, buf); }
void GetDoubleArrayRegion(jdoubleArray array, jsize start, jsize len,jdouble* buf)
{ functions->GetDoubleArrayRegion(this, array, start, len, buf); }
```

在 jni 预先开辟一段内存，然后把 Java 数组拷贝到这段内存中; 该方法有四个参数，第一个指向的是 java 数组，第二个参数表示开始位置，第三个参数表示拷贝的长度，第四个指向目标缓冲区；

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jintArray intArray){
    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");

    jsize size = env->GetArrayLength(intArray);
    jint* data = new jint[size];
    env->GetIntArrayRegion(intArray, 0, size, data);
    for (int j = 0; j < size; j++) {
        __android_log_print(ANDROID_LOG_INFO, "native", "data %d", data[j]);
    }

    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

5. Set<Type>ArrayRegion(<Type>Array arr , jsize start , jsize len , const <Type>* buffer)

```
void SetBooleanArrayRegion(jbooleanArray array, jsize start, jsize len,const jboolean* buf)
{ functions->SetBooleanArrayRegion(this, array, start, len, buf); }
void SetByteArrayRegion(jbyteArray array, jsize start, jsize len,const jbyte* buf)
{ functions->SetByteArrayRegion(this, array, start, len, buf); }
void SetCharArrayRegion(jcharArray array, jsize start, jsize len,const jchar* buf)
{ functions->SetCharArrayRegion(this, array, start, len, buf); }
void SetShortArrayRegion(jshortArray array, jsize start, jsize len,const jshort* buf)
{ functions->SetShortArrayRegion(this, array, start, len, buf); }
void SetIntArrayRegion(jintArray array, jsize start, jsize len,const jint* buf)
```

```
{ functions->SetIntArrayRegion(this, array, start, len, buf); }
void SetLongArrayRegion(jlongArray array, jsize start, jsize len,const jlong* buf)
{ functions->SetLongArrayRegion(this, array, start, len, buf); }
void SetFloatArrayRegion(jfloatArray array, jsize start, jsize len,const jfloat* buf)
{ functions->SetFloatArrayRegion(this, array, start, len, buf); }
void SetDoubleArrayRegion(jdoubleArray array, jsize start, jsize len,const jdouble* buf)
{ functions->SetDoubleArrayRegion(this, array, start, len, buf); }
```

把 Java 数组中的指定范围的元素用 jni 数组中的元素来赋值,该方法有四个参数，第一个指向的是目标 java 数组，第二个参数表示开始位置，第三个参数表示拷贝的长度，第四个指向 jni 数组；

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jintArray intArray){
    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");

    jsize size = env->GetArrayLength(intArray);
    jint* data = new jint[size];
    for (int j = 0; j < size; j++) {
        data[j] = j;
    }
    env->SetIntArrayRegion(intArray, 0, size, data);

    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

6. <Type>Array New<Type>Array(jsize sz)

```
jbooleanArray NewBooleanArray(jsize length)
{ return functions->NewBooleanArray(this, length); }
jbyteArray NewByteArray(jsize length)
{ return functions->NewByteArray(this, length); }
jcharArray NewCharArray(jsize length)
{ return functions->NewCharArray(this, length); }
jshortArray NewShortArray(jsize length)
{ return functions->NewShortArray(this, length); }
jintArray NewIntArray(jsize length)
{ return functions->NewIntArray(this, length); }
jlongArray NewLongArray(jsize length)
{ return functions->NewLongArray(this, length); }
jfloatArray NewFloatArray(jsize length)
{ return functions->NewFloatArray(this, length); }
jdoubleArray NewDoubleArray(jsize length)
{ return functions->NewDoubleArray(this, length); }
```

使用 NewBooleanArray(jsize length) 可以返回一个指定长度的 java 数组；

7. jobject GetObjectArrayElement(jobjectArray array, jsize index) 与 void SetObjectArrayElement(jobjectArray array, jsize index, jobject value)

上面所述的对于数组的操作除了 GetArrayLength 方法是对于基本类型数组与对象数组通用的，其它均是用于操作基本类型数组；GetObjectArrayElement 与 SetObjectArrayElement 提供了对于对象数组的操作 GetObjectArrayElement 方法用于返回对象数组中指定位置的对象；SetObjectArrayElement 用于更新对象数组中指定位置的对象；

8. jobjectArray NewObjectArray(jsize length, jclass elementClass, jobject initialElement)

之前提到 New<Type>Array 方法用于创建 java 基本类型数组，对应的 NewObjectArray 用于创建 java 对象数组，该方法包含三个参数，第一个参数表示数组长度，第二个参数表示

类，第三个参数表示元素初始值；