JNI 字符串操作

efan 关注

0.168 2018.06.11 21:32:17 字数 1,014 阅读 2,035

1. NewString

```
jstring NewString(const jchar* unicodeChars, jsize len)
{ return functions->NewString(this, unicodeChars, len); }
```

该方法会利用(Unicode) char 数组生成并返回 java String对象；

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

    jchar* data = new jchar[5];
    data[0] = 'h';
    data[1] = 'e';
    data[2] = 'l';
    data[3] = 'l';
    data[4] = 'o';
    return env->NewString(data, 5);
}
```

2. GetStringLength

```
jsize GetStringLength(jstring string)
{ return functions->GetStringLength(this, string); }
```

返回 java 字符串的长度(Unicode)

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

    jint length = env->GetStringLength(s);
    __android_log_print(ANDROID_LOG_INFO, "native", "string length:%d", length);
    return s;
}
```

3. GetStringChars 与 ReleaseStringChars

```
const jchar* GetStringChars(jstring string, jboolean* isCopy)
{ return functions->GetStringChars(this, string, isCopy); }

void ReleaseStringChars(jstring string, const jchar* chars)
{ functions->ReleaseStringChars(this, string, chars); }
```

GetStringChars 方法用于返回一个对应 java String 对象的(Unicode) char 数组指针，返回的指针可能指向 java String 对象，也可能是指向 jni 中的拷贝，参数 isCopy 用于返回是否是拷贝；

ReleaseStringChars 方法用于释放与 java String 对象绑定的(Unicode) char 数组指针；

因为 GetStringChars 方法的返回值用 const 修饰，所以获取的(Unicode) char 数组是不能被更改的；

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

    jboolean isCopy = JNI_FALSE;
    jsize length = env->GetStringLength(s);
    const jchar* chars = env->GetStringChars(s, &isCopy);
    for(int i = 0; i < length; i++){
        __android_log_print(ANDROID_LOG_INFO, "native", "char %c", chars[i]);
    }
    env->ReleaseStringChars(s, chars);
    return s;
}
```

4. NewString(Unicode)

```
jstring NewStringUTF(const char* bytes)
{ return functions->NewStringUTF(this, bytes); }
```

该方法会利用(UTF-8) char 数组生成并返回 java String对象；

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

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

5. GetStringUTFLength

```
jsize GetStringUTFLength(jstring string)
{ return functions->GetStringUTFLength(this, string); }
```

返回 java String 的长度(UTF-8)；

```
jstring stringFromJNI(JNIEnv *env, jobject thiz, jstring s){
    __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");
    jint length = env->GetStringUTFLength(s);
    __android_log_print(ANDROID_LOG_INFO, "native", "length: %d", length);
    return s;
}
```

6. GetStringUTFChars 与 ReleaseStringUTFChars

```
const char* GetStringUTFChars(jstring string, jboolean* isCopy)
{ return functions->GetStringUTFChars(this, string, isCopy); }

void ReleaseStringUTFChars(jstring string, const char* utf)
{ functions->ReleaseStringUTFChars(this, string, utf); }
```

GetStringChars 方法用于返回一个对应 java String 对象的(UTF-8) char 数组指针，返回的指针可能指向 java String 对象，也可能是指向 jni 中的拷贝，参数 isCopy 用于返回是否是拷

贝；

ReleaseStringChars 方法用于释放与 java String 对象绑定的(UTF-8) char 数组指针；

因为 GetStringChars 方法的返回值用 const 修饰，所以获取的(UTF-8) char 数组是不能被更改的；

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

    jboolean isCopy = JNI_FALSE;
    jsize length = env->GetStringUTFLength(s);
    const char* chars = env->GetStringUTFChars(s, &isCopy);
    for(int i = 0; i < length; i++){
        __android_log_print(ANDROID_LOG_INFO, "native", "char %c", chars[i]);
    }
    env->ReleaseStringUTFChars(s, chars);
    return s;
}
```

7. GetStringRegion

```
void GetStringRegion(jstring str, jsize start, jsize len, jchar* buf)
{ functions->GetStringRegion(this, str, start, len, buf); }
```

拷贝 java String 指定范围的字符至 jni 本地(Unicode)；

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

    jchar* chars = new jchar[5];
    env->GetStringRegion(s, 0, 5, chars);
    return env->NewString(chars, 5);
}
```

8. GetStringUTFRegion

```
void GetStringUTFRegion(jstring str, jsize start, jsize len, char* buf)
{ return functions->GetStringUTFRegion(this, str, start, len, buf); }
```

拷贝 java String 指定范围的字符至 jni 本地(UTF-8)；

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

    char* chars = new char[5];
    env->GetStringUTFRegion(s, 0, 5, chars);
    return env->NewStringUTF(chars);
}
```

9. GetStringCritical 与 ReleaseStringCritical

```
void* GetPrimitiveArrayCritical(jarray array, jboolean* isCopy)
{ return functions->GetPrimitiveArrayCritical(this, array, isCopy); }


```

```
5 void ReleasePrimitiveArrayCritical(jarray array, void* carray, jint mode)
  { functions->ReleasePrimitiveArrayCritical(this, array, carray, mode); }
```

GetStringCritical、ReleaseStringCritical 与 GetStringChars、ReleaseStringChars方法非常相似；区别在于前者更倾向于获取 java String 的指针，而不是进行拷贝；

GetStringCritical 方法用于返回一个对应 java String 对象的(Unicode) char 数组指针，返回的指针可能指向 java String 对象，也可能是指向 jni 中的拷贝，参数 isCopy 用于返回是否是拷贝；

ReleaseStringCritical 方法用于释放与 java String 对象绑定的(Unicode) char 数组指针；

因为 ReleaseStringCritical 方法的返回值用 const 修饰，所以获取的(Unicode) char 数组是不能被更改的；

```
1  jstring stringFromJNI(JNIEnv *env, jobject thiz, jstring s){
2      __android_log_print(ANDROID_LOG_INFO, "native", "stringFromJNI");
3
4      jboolean isCopy = JNI_FALSE;
5      jsize length = env->GetStringLength(s);
6      const jchar* chars = env->GetStringCritical(s, &isCopy);
7      __android_log_print(ANDROID_LOG_INFO, "native", "isCopy %d", isCopy);
8      for(int i = 0; i < length; i++){
9          __android_log_print(ANDROID_LOG_INFO, "native", "char %c", chars[i]);
10     }
11     env->ReleaseStringCritical(s, chars);
12     return s;
13 }
```