

Jni中C++和Java的参数传递

如何使用JNI的一些基本方法和过程在网上多如牛毛，如果你对Jni不甚了解，不知道Jni是做什么的，如何建立一个基本的jni程序，或许可以参考下面这些文章：

[利用VC++ 6.0实现JNI的最简单的例子](#)

[JNI入门教程之HelloWorld篇](#)

[SUN JNI Tutorial](#)

这些资料的例子中，大多数只是输入一些简单的参数，获取没有参数。而在实际的使用过程中，往往需要对参数进行处理转换。才可以被C/C++程序识别。比如我们在C++中有一个结构（Struct）DiskInfo，需要传递一个类似于DiskInfo *pDiskInfo的参数，类似于在C++中这样参数如何传递到Java中呢？下面我们就来讨论C++到Java中方法的一些常见参数的转换：

定义Native Java类：

如果你习惯了使用JNI，你就不会觉得它难了。既然本地方法是由其他语言实现的，它们在Java中没有函数体。但是，所有本地代码必须用本地关键词声明，成为Java类的成员。假设我们在C++中有这么一个结构，它用来描述硬盘信息：

```
// 硬盘信息
struct {
    char name[ 256 ];
    int serial;
} DiskInfo;
```

那么我们需要在Java中定义一个类来与之匹配，声明可以写成这样：

```
class DiskInfo {
    // 名字
    public String name;

    // 序列号
    public int serial;
}
```

在这个类中，申明一些Native的本地方法，来测试方法参数的传递，分别定义了一些函数，用来传递结构或者结构数组，具体定义如下面代码：

```
/* ***** 定义本地方法 ***** */
// 输入常用的数值类型(Boolean,Byte,Char,Short,Int,Float,Double)
public native void displayParms(String showText, int i, boolean bl);

// 调用一个静态方法
public native int add( int a, int b);

// 输入一个数组
public native void setArray(boolean[] blList);

// 返回一个字符串数组
public native String[] getStringArray();

// 返回一个结构
public native DiskInfo getStruct();

// 返回一个结构数组
public native DiskInfo[] getStructArray();
```

编译生成C/C++头文件

定义好了Java类之后，接下来就要写本地代码。本地方法符号提供一个满足约定的头文件，使用Java工具Javah可以很容易地创建它而不用手动去创建。你对Java的class文件使用javah命令，就会为你生成一个对应的C/C++头文件。

1、在控制台下进入工作路径，本工程路径为：E:\work\java\workspace\JavaJni。

2、运行javah 命令：javah -classpath E:\work\java\workspace\JavaJni com.sundy.jnidemo ChangeMethodFromJni

本文生成的C/C++头文件名为：com_sundy_jnidemo_ChangeMethodFromJni.h

在C/C++中实现本地方法

生成C/C++头文件之后，你就需要写头文件对应的本地方法。注意：所有的本地方法的第一个参数都是指向JNIEnv结构的。这个结构是用来调用JNI函数的。第二个参数jclass的意义，要看方法是不是静态的（static）或者实例（Instance）的。前者，jclass代表一类对象的引用，而后者是被调用的方法所属对象的引用。

返回值和参数类型根据等价位约定映射到本地C/C++类型，如表JNI类型映射所示。有些类型，在本地代码中可直接使用，而其他类型只有通过JNI调用操作。

表A

<	201	
日	一	二
24	25	26
1	2	3
8	9	10
15	16	17
22	23	24
29	30	31

导航

[博客园](#)

[首页](#)

[新随笔](#)

[联系](#)

[订阅](#) [XML](#)

[管理](#)

统计

[随笔](#) - 6

[文章](#) - 21

[评论](#) - 5

[引用](#) - 0

公告

昵称：and_tt

园龄：6年2个月

粉丝：19

关注：0

[+加关注](#)

搜索

常用链接

[我的随笔](#)

[我的评论](#)

[我的参与](#)

[最新评论](#)

[我的标签](#)

我的标签

[android\(2\)](#)

[Thinking\(2\)](#)

[类型转换\(2\)](#)

[数据结构\(1\)](#)

[文件系统\(1\)](#)

[ubuntu\(1\)](#)

[ubuntu安装\(1\)](#)

[wince\(1\)](#)

[fat\(1\)](#)

[jni\(1\)](#)

[更多](#)

随笔档案

[2012年2月 \(2\)](#)

[2011年9月 \(1\)](#)

[2011年8月 \(2\)](#)

[2011年7月 \(1\)](#)

文章分类

[.net](#)

[android\(8\)](#)

[c/c++\(1\)](#)

[hardware](#)

[java\(1\)](#)

[linux\(7\)](#)

[others\(3\)](#)

[qt](#)

[scm](#)

[wince](#)

[zigbee](#)

最新评论

1. Re: (转)linux下

android一个简单jni

“百度经验”上有一篇

编译 hello jni”的文

的，里面还提到了新

一系列的问题，希望

2. Re: (转)ubuntu中

原型

博客版面很有个性，

Java 类型	本地类型	描述
boolean	jboolean	C/C++8位整型
byte	jbyte	C/C++带符号的8位整型
char	jchar	C/C++无符号的16位整型
short	jshort	C/C++带符号的16位整型
int	jint	C/C++带符号的32位整型
long	jlong	C/C++带符号的64位整型e
float	jfloat	C/C++32位浮点型
double	jdouble	C/C++64位浮点型
Object	jobject	任何Java对象，或者没有对应java类型的对象
Class	jclass	Class对象
String	jstring	字符串对象
Object[]	jobjectArray	任何对象的数组
boolean[]	jbooleanArray	布尔型数组
byte[]	jbyteArray	比特型数组
char[]	jcharArray	字符型数组
short[]	jshortArray	短整型数组
int[]	jintArray	整型数组
long[]	jlongArray	长整型数组
float[]	jfloatArray	浮点型数组
double[]	jdoubleArray	双浮点型数组

※ JNI类型映射

使用数组：

JNI通过JNIEnv提供的操作Java数组的功能。它提供了两个函数：一个是操作java的简单型数组的，另一个是操作对象类型数组的。

因为速度的原因，简单类型的数组作为指向本地类型的指针暴露给本地代码。因此，它们能作为常规的数组存取。这个指针是指向实际的Java数组或者Java数组的拷贝的指针。另外，数组的布置保证匹配本地类型。

为了存取Java简单类型的数组，你就要使用GetXXXArrayElements函数（见表B），XXX代表了数组的类型。这个函数把Java数组看成参数，返回一个指向对应的本地类型的数组的指针。

表B

函数	Java 数组类型	本地类型
GetBooleanArrayElements	jbooleanArray	jboolean
GetByteArrayElements	jbyteArray	jbyte
GetCharArrayElements	jcharArray	jchar
GetShortArrayElements	jshortArray	jshort
GetIntArrayElements	jintArray	jint
GetLongArrayElements	jlongArray	jlong
GetFloatArrayElements	jfloatArray	jfloat
GetDoubleArrayElements	jdoubleArray	jdouble

JNI数组存取函数

当你对数组的存取完成后，要确保调用相应的ReleaseXXXArrayElements函数，参数是对应Java数组和GetXXXArrayElements返回的指针。如果必要的话，这个释放函数会复制你做的任何变化（这样它们就反射到java数组），然后释放所有相关的资源。

为了使用java对象的数组，你必须使用GetObjectArrayElement函数和SetObjectArrayElement函数，分别去get，set数组的元素。GetArrayLength函数会返回数组的长度。

使用对象

JNI提供的另外一个功能是在本地代码中使用Java对象。通过使用合适的JNI函数，你可以创建Java对象，get、set 静态(static)和实例(instance)的域，调用静态(static)和实例(instance)函数。JNI通过ID识别域和方法，一个域或方法的ID是任何处理域和方法的函数的必须参数。

表C列出了用以得到静态(static)和实例(instance)的域与方法的JNI函数。每个函数接受（作为参数）域或方法的类，它们的名称，符号和它们对应返回的jfieldID或jmethodID。

表C

函数	描述
GetFieldID	得到一个实例的域的ID
GetStaticFieldID	得到一个静态的域的ID
GetMethodID	得到一个实例的方法的ID
GetStaticMethodID	得到一个静态方法的ID

※域和方法的函数

如果你有了一个类的实例，它就可以通过方法GetObjectClass得到，或者如果你没有这个类的实例，可以通过FindClass得到。符号是从域的类型或者方法的参数，返回值得到字符串，如表D所示。

表D

3. Re:fcntl使用dup2函数那里是不

4. Re:FAT文件系统不错

5. Re:fcntl使用@@

阅读排行榜

- 1. linux 下串口编程(5384)
- 2. 由进程名称获取窗
- 3. ubuntu自动加载
- 4. 安装ubuntu11.0(383)
- 5. wince编程CStrin

推荐排行榜

- 1. linux 下串口编程(1)

Java 类型	符号
boolean	Z
byte	B
char	C
short	S
int	I
long	L
float	F
double	D
void	V
objects对象	Lfully-qualified-class-name;L类名
Arrays数组	[array-type [数组类型
methods方法	(argument-types)return-type(参数类型)返回类型

※确定域和方法的符号

下面我们来看看，如果通过使用数组和对象，从C++中的获取到Java中的DiskInfo 类对象，并返回一个DiskInfo数组：

```
//返回一个结构数组，返回一个硬盘信息的结构数组
JNIEXPORT jobjectArray JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_getStructArray
(JNIEnv *env, jobject _obj)
{
    //申明一个object数组
    jobjectArray args = 0;

    //数组大小
    jsize len = 5;

    //获取object所属类,一般为java/lang/Object就可以了
    jclass objClass = (env)->FindClass("java/lang/Object");

    //新建object数组
    args = (env)->NewObjectArray(len, objClass, 0);

    /* 下面为获取到Java中对应的实例类中的变量*/

    //获取Java中的实例类
    jclass objectClass = (env)->FindClass("com/sundy/jnidemo/DiskInfo");

    //获取类中每一个变量的定义
    //名字
    jfieldID str = (env)->GetFieldID(objectClass,"name","Ljava/lang/String;");
    //序号
    jfieldID ival = (env)->GetFieldID(objectClass,"serial","I");

    //给每一个实例的变量付值，并且将实例作为一个object，添加到objcet数组中
    for(int i=0; i < len; i++ )
    {
        //给每一个实例的变量付值
        jstring jstr = WindowsTojstring(env,"我的磁盘名字是 D:");
        //(env)->SetObjectField(_obj,str,(env)->NewStringUTF("my name is D:"));
        (env)->SetObjectField(_obj,str,jstr);
        (env)->SetShortField(_obj,ival,10);

        //添加到objcet数组中
        (env)->SetObjectArrayElement(args, i, _obj);
    }
    //返回object数组
    return args;
}
```

全部的C/C++方法实现代码如下：

```
/*
 *
 * 一缕阳光(sundy)版权所有，保留所有权利。
 */
/**
 *
 *  TODO Jni 中一个从Java到C/C++参数传递测试类
 */
```

```

* @author 刘正伟(sundy)
* @see http://www.cnweblog.com/sundy
* @see mailto:sundy26@126.com
* @version 1.0
* @since 2005-4-30
*
* 修改记录 :
*
* 日期      修改人      描述
* -----
*
*
*
*/
// JniManage.cpp : 定义 DLL 应用程序的入口点。
//
package com.sundy.jnidemo;
#include "stdafx.h"

#include <stdio.h>
#include <math.h>
#include "jni.h"
#include "jni_md.h"

#include "../head/Base.h"
#include "head/wmi.h"
#include "head/com_sundy_jnidemo_ChangeMethodFromJni.h" //通过javah -jni javactransfer 生成
#include <stdio.h>
#include "stdlib.h"
#include "string.h"

#pragma comment (lib,"BaseInfo.lib")
#pragma comment (lib,"jvm.lib")
//硬盘信息
struct {
    char name[256];
    int serial;
}DiskInfo;
/*BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                       )
{
    LPTSTR strName = new CHAR[256] ;
    (*GetHostName)(strName);
    printf("%s\n",strName);
    delete [] strName;

    return TRUE;
}*/
//将jstring类型转换成windows类型
char* jstringToWindows( JNIEnv *env, jstring jstr );
//将windows类型转换成jstring类型
jstring WindowsToJstring( JNIEnv* env, char* str );

//主函数
BOOL WINAPI DllMain(HANDLE hHandle, DWORD dwReason, LPVOID lpReserved)
{
    return TRUE;
}
//输入常用的数值类型 Boolean,Byte,Char,Short,Int,Float,Double
JNIEXPORT void JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_displayParms
(JNIEnv *env, jobject obj, jstring s, jint i, jboolean b)
{
    const char* szStr = (env)->GetStringUTFChars(s, 0 );
    printf( "String = [%s]\n", szStr );
    printf( "int = %d\n", i );
    printf( "boolean = %s\n", (b==JNI_TRUE ? "true" : "false") );
    (env)->ReleaseStringUTFChars(s, szStr );
}

//调用一个静态方法,只有一个简单类型输出
JNIEXPORT jint JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_add
(JNIEnv *env, jobject, jint a, jint b)

```

```
    {
        int rtn = (int)(a + b);
        return (jint)rtn;
    }

    ///输入一个数组,这里输入的是一个Boolean类型的数组
    JNIEXPORT void JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_setArray
    (JNIEnv *env, jobject, jbooleanArray ba)
    {
        jboolean* pba = (env)->GetBooleanArrayElements(ba, 0 );
        jsize len = (env)->GetArrayLength(ba);
        int i=0;
        // change even array elements
        for( i=0; i < len; i+=2 )
        {
            pba[i] = JNI_FALSE;
            printf( "boolean = %s\n", (pba[i]==JNI_TRUE ? "true" : "false") );
        }
        (env)->ReleaseBooleanArrayElements(ba, pba, 0 );
    }

    ///返回一个字符串数组
    JNIEXPORT jobjectArray JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_getStringArray
    (JNIEnv *env, jobject)
    {
        jstring str;
        jobjectArray args = 0;
        jsize len = 5;
        char* sa[] = { "Hello,", "world!", "JNI", "is", "fun" };
        int i=0;
        args = (env)->NewObjectArray(len,(env)->FindClass("java/lang/String"),0);
        for( i=0; i < len; i++ )
        {
            str = (env)->NewStringUTF(sa[i] );
            (env)->SetObjectArrayElement(args, i, str);
        }
        return args;
    }

    ///返回一个结构,这里返回一个硬盘信息的简单结构类型
    JNIEXPORT jobject JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_getStruct
    (JNIEnv *env, jobject obj)
    {
        /* 下面为获取到Java中对应的实例类中的变量*/

        //获取Java中的实例类
        jclass objectClass = (env)->FindClass("com/sundy/jnidemo/DiskInfo");

        //获取类中每一个变量的定义
        //名字
        jfieldID str = (env)->GetFieldID(objectClass,"name","Ljava/lang/String;");
        //序号
        jfieldID ival = (env)->GetFieldID(objectClass,"serial","I");

        //给每一个实例的变量赋值
        (env)->SetObjectField(obj,str,(env)->NewStringUTF("my name is D:"));
        (env)->SetShortField(obj,ival,10);

        return obj;
    }

    ///返回一个结构数组, 返回一个硬盘信息的结构数组
    JNIEXPORT jobjectArray JNICALL Java_com_sundy_jnidemo_ChangeMethodFromJni_getStructArray
    (JNIEnv *env, jobject _obj)
    {
        //申明一个object数组
        jobjectArray args = 0;

        //数组大小
        jsize len = 5;

        //获取object所属类,一般为java/lang/Object就可以了
        jclass objClass = (env)->FindClass("java/lang/Object");
    }
```

```

//新建object数组
args = (env)->NewObjectArray(len, objClass, 0);

/* 下面为获取到Java中对应的实例类中的变量*/

//获取Java中的实例类
jclass objectClass = (env)->FindClass("com/sundy/jnidemo/DiskInfo");

//获取类中每一个变量的定义
//名字
jfieldID str = (env)->GetFieldID(objectClass,"name","Ljava/lang/String;");
//序号
jfieldID ival = (env)->GetFieldID(objectClass,"serial","I");

//给每一个实例的变量赋值，并且将实例作为一个object，添加到objcet数组中
for(int i=0; i < len; i++ )
{
    //给每一个实例的变量赋值
    jstring jstr = WindowsToJstring(env,"我的磁盘名字是 D:");
    //(env)->SetObjectField(_obj,str,(env)->NewStringUTF("my name is D:"));
    (env)->SetObjectField(_obj,str,jstr);
    (env)->SetShortField(_obj,ival,10);

    //添加到objcet数组中
    (env)->SetObjectArrayElement(args, i, _obj);
}
//返回object数组
return args;
}

//将jstring类型转换成windows类型
char* jstringToWindows( JNIEnv *env, jstring jstr )
{
    int length = (env)->GetStringLength(jstr );
    const jchar* jcstr = (env)->GetStringChars(jstr, 0 );
    char* rtn = (char*)malloc( length*2+1 );
    int size = 0;
    size = WideCharToMultiByte( CP_ACP, 0, (LPCWSTR)jcstr, length, rtn,(length*2+1), NULL, NULL );
    if( size <= 0 )
        return NULL;
    (env)->ReleaseStringChars(jstr, jcstr );
    rtn[size] = 0;
    return rtn;
}

//将windows类型转换成jstring类型
jstring WindowsToJstring( JNIEnv* env, char* str )
{
    jstring rtn = 0;
    int slen = strlen(str);
    unsigned short * buffer = 0;
    if( slen == 0 )
        rtn = (env)->NewStringUTF(str );
    else
    {
        int length = MultiByteToWideChar( CP_ACP, 0, (LPCSTR)str, slen, NULL, 0 );
        buffer = (unsigned short *)malloc( length*2 + 1 );
        if( MultiByteToWideChar( CP_ACP, 0, (LPCSTR)str, slen, (LPWSTR)buffer, length ) >0 )
            rtn = (env)->NewString( (jchar*)buffer, length );
    }
    if( buffer )
        free( buffer );
    return rtn;
}

```

Java 测试native代码

这没有什么多说的，看代码吧

```

//主测试程序
public static void main(String[] args) {
    ChangeMethodFromJni changeJni = new ChangeMethodFromJni();

    //输入常用的数值类型(string int boolean)

```

```
System.out
.println("-----输入常用的数值类型(string int boolean)-----");
changeJni.displayParms("Hello World!", 100, true);

//调用一个静态方法
System.out.println("-----调用一个静态方法-----");
int ret = changeJni.add(12, 20);
System.out.println("The result is: " + String.valueOf(ret));

//输入一个数组
System.out.println("-----输入一个数组-----");
boolean[] blList = new boolean[] { true, false, true };
changeJni.setArray(blList);

//返回一个字符串数组
System.out.println("-----返回一个字符串数组-----");
String[] strList = changeJni.getStringArray();
for (int i = 0; i < strList.length; i++) {
    System.out.print(strList[i]);
}
System.out.println();

System.out.println("-----返回一个结构-----");

//返回一个结构
DiskInfo disk = changeJni.getStruct();
System.out.println("name:" + disk.name);
System.out.println("Serial:" + disk.serial);

//返回一个结构数组

System.out.println("-----返回一个结构数组 -----");
DiskInfo[] diskList = changeJni.getStructArray();
for (int i = 0; i < diskList.length; i++) {
    System.out.println("name:" + diskList[i].name);
    System.out.println("Serial:" + diskList[i].serial);
}
}
```

原文：<http://www.blogjava.net/china-qd/archive/2006/04/29/44002.html>

分类: android

标签: jni

好文要顶

关注我

收藏该文



and_tt
关注 - 0
粉丝 - 19

+加关注

« 上一篇: start

» 下一篇: linux 下串口编程VTIME和VMIN的设置

posted on 2011-08-19 12:43 and_tt 阅读(7734) 评论(0) 编辑 收藏

0

推荐

0

反对

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】腾讯云 十分钟定制你的第一个微信小程序

【推荐】报表开发有捷径：快速设计轻松集成，数据可视化和交互

Powered by:

博客园

Copyright © and_tt