Java可以调用C语言，即调用本地代码生成的dll文件或者so文件。如何生成dll和so文件，已经在C语言中学过了。这里不再说明。

Java调用dll、so文件，使用JNI技术，但非常麻烦，因此我们这里使用JNA框架，此开源框架的地址是：https://github.com/java-native-access/jna。

编写C语言代码如下：

|  |
| --- |
| #include <stdio.h>  \_\_declspec(dllexport)  int sum(int a, int b)  {  printf("The C Program will return : %d\n", a + b);  return a + b;  } |

编译生成x64位的dll文件，注意，生成dll文件的位数要和JDK的位数一致，否则以后的java程序会出现“UnsatisfiedLinkError: %1 不是有效的 Win32 应用程序。”的异常。

现在在项目中引入好JNA需要的jar包（参见上述的网址），然后我们把生成的“CAD.dll”文件放入Java项目的src目录下，首先编写一个接口：

|  |
| --- |
| **package** juc.demo;  **import** com.sun.jna.Library; **import** com.sun.jna.Native;  **public interface** CAD **extends** Library {  CAD ***instanceDll*** = (CAD) Native.*loadLibrary*(**"CAD"**,CAD.**class**); // 加载dll。参数1是dll文件名  **public int** sum(**int** a,**int** b); // dll中对应的方法。Java与C语言的数据类型对应关系还是参照GitHub。 } |

然后就能直接调用了：

|  |
| --- |
| **package** juc.demo;  **public class** TestDLL {  **public static void** main(String[] args) {  **int** sum = CAD.***instanceDll***.sum(1, 2);  System.***out***.println(sum);  } } |

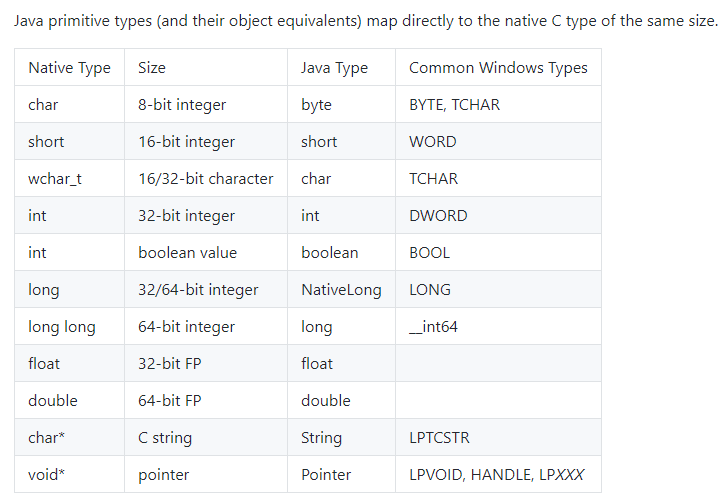
Java输出的结果是：

|  |
| --- |
| 3  The C Program will return : 3 |

其中下面一句话是C程序输出的。

重点问题是Java与C数据类型是如何映射的。这里讲几个常用的。

在Github项目官网中，https://github.com/java-native-access/jna#using-the-library这一段讲了如何映射。比如“Mapping between Java and Native”一文中：



比如在C语言中使用char\*：

|  |
| --- |
| \_\_declspec(dllexport)  char\* testStr(char \* ch)  {  return ch;  } |

那么我们在Java接口中模拟的方法原型为（就是String）：

|  |
| --- |
| String testStr(String ch); |

调用是没问题的。

下面我们讲讲数组和结构体的使用。

（1）对于C语言中的数组，对应的Java中也是数组。当数组作为参数时，那么在C语言中最好再加上一个length参数表示数组的长度，以便正常遍历：

|  |
| --- |
| \_\_declspec(dllexport)  void testArray(int arrays[], int length)  {  for (int i = 0; i < length; i++)  {  printf("%d\n", arrays[i]);  }  } |

那么Java中对应接口为：

|  |
| --- |
| **void** testArray(**int**[] arrays, **int** length); |

（2）对于结构体的模拟还是稍微复杂点的。比如下面C函数返回了一个STUDENT结构体：

|  |
| --- |
| typedef struct Student  {  char\* name;  int age;  }STUDENT;  \_\_declspec(dllexport)  STUDENT testStruct()  {  STUDENT student;  student.name = "Jack";  student.age = 12;  return student;  } |

方法返回的是结构体。那么Java中如何模拟结构体呢？

首先我们要写一个类（比如就叫Student）继承Structure，在类中按顺序写好结构体中的属性（公开的字段），并且在getFieldOrder方法中返回按顺序的字段名称List集合。并且，还类中还写了两个空的内部类ByReference和ByValue（稍后知道这是用来标记是值还是引用的）。

|  |
| --- |
| **package** juc.demo;  **import** com.sun.jna.Structure; **import** java.util.Arrays; **import** java.util.List;  **public class** Student **extends** Structure{  **public static class** ByReference **extends** Student **implements** Structure.ByReference{  }   **public static class** ByValue **extends** Student **implements** Structure.ByValue{  }   **public** String **name**;  **public int age**;   @Override  **protected** List<String> getFieldOrder() {  **return** Arrays.*asList*(**"name"**, **"age"**);  } } |

此时接口中内容为：

|  |
| --- |
| **package** juc.demo;  **import** com.sun.jna.Library; **import** com.sun.jna.Native;  **public interface** CAD **extends** Library {  CAD ***instanceDll*** = (CAD) Native.*loadLibrary*(**"ConsoleApplication2"**,CAD.**class**);  Student.ByValue testStruct(); } |

注意testStruct()返回值是Student.ByValue。那么我们在main方法中就能使用了：

|  |
| --- |
| **public class** TestDLL {  **public static void** main(String[] args) {  Student.ByValue stu = CAD.***instanceDll***.testStruct();  System.***out***.println(stu.**name**);  System.***out***.println(stu.**age**);  } } |

至于为啥用ByValue或者ByReference，详见网址：

http://blog.csdn.net/xie\_heng/article/details/49814207

和：

http://blog.csdn.net/shendl/article/details/3599849

结束。