Java本地接口工作方式初探 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022_Java_E6_9C_AC_E5_9C_B0_c104_252297.htm Java本地接口(Java Native Interface (JNI))允许运行在Java虚拟机(Java Virtual Machine (JVM))上的代码调用本地程序和类库，或者被它们调用，这些程序和类库可以是其它语言编写的，比如C、C 或者汇编语言。 当一个程序无法完全使用Java编写时，开发者可以通过JNI来编写本地方法，比如标准Java类库并不支持的依赖于平台的特色或者程序库。JNI还可以用于修改现有的使用其它语言编写的程序，使它们可以通过Java编写的程序来访问。很多基本类库都依赖JNI来为开发者和用户提供服务，比如文件的输入/输出和音频功能。在基本类库中包含的对于性能和平台敏感的API可以允许所有的Java程序以安全和平台无关的方式来使用这些功能，在采用JNI之前，开发者需要明确这些功能并不是已经包含在Java标准类库中的，在这篇文章中，我将会讲解JNI是如何工作的以及本地类型是如何映射到Java的类型和类库的。JNI工作原理 在JNI中，本地函数是通过一个独立的.c或.cpp文件来实现的(C 为JNI提供的界面会更简洁一些)。当JVM调用该函数时，它传递了一个JNIEnv指针、一个jobject指针和通过Java方法定义的Java参数，JNI函数的形式如下：JNIEXPORT void JNICALL Java_ClassName_MethodName (JNIEnv *env, jobjectobj){ //Method native implemenation} env指针是一个包含了JVM接口的结构，它包含了与JVM进行交互以及与Java对象协同工作所必需的函数，示例中的JNI函数可以在本地数组和Java数组类

型之间、本地字符串和Java字符串类型之间进行转换，其功能还包括对象的实例化、抛出异常等。基本上您可以使用JNIEnv来实现所有Java能做到的事情，虽然要简单很多。更加正式的解释是这样的，本地代码通过调用JNI的函数来访问JVM，这是通过一个界面指针实现的（界面指针实际上是指向指针的指针），该指针指向一个指针数组，数组中的每个指针都指向了一个界面函数，而每个界面函数都是在数组中预先定义过的。 本地方法将JNI界面指针当作一个参数，如果在同一个Java线程中，出现对该本地方法的多重调用，JVM则保证传递相同的界面指针到本地方法。不过，一个本地方法可以被不同的Java线程调用，因而也可能会收到不同的JNI界面指针。 本地方法是通过System.loadLibrary方法加载的，在以下的例子中，类的初始化方法加载了一个指定平台的本地类库，该类库定义了本地方法：packagepkg. class Cls { native double f(inti, String s). static { System.loadLibrary(pkg_Cls"). } } System.loadLibrary方法的参数是一个类库的名称，它可以由程序员任意选取，系统则遵循一个标准的本地化平台的方式来转换类库的名称到一个本地类库的名称。例如，在Solaris操作系统中会将pkg_Cls转换为libpkg_Cls.so，而Win32系统则会将同样的pkg_Cls转换为pkg_Cls.dll。 动态指针会根据它们的名字来进行解析，一个本地方法的名称是按照组件进行连接的，它包含了：前缀“ Java_ ”、一个分离的合法的类名称和一个分离的方法名称。 注意：微软的JVM有相同的机制从Java调用本地Windows代码，该机制被称为原始本地接口(Raw Native Interface (RNI))。数据类型映射 基本类型，比如整型、字符等等，是在Java和本地代码间进行拷贝的，而其他的自定

义Java对象则是通过引用来传递的。 这个表格展示了Java和本地代码之间的类型映射，这些类型是可以互换的，您可以在您使用int类型的位置使用jint类型，当然反过来也一样，而且不需要任何类型转化。但是，Java的字符串和数组类型和本地的字符串与数组类型之间的转换就比较困难了，如果您使用的jstring类型中出现了字符“ * ”，您的代码会造成JVM的崩溃，以下的例子说明了您应当如何正确使用字符串：JNIEXPORT void JNICALL Java_ClassName_MethodName (JNIEnv *env, jobjectobj, jstringjavaString){ //Get the native string from Java string const char *nativeString = env->GetStringUTFChars(env,javaString, 0). printf("%s", nativeString). env->ReleaseStringUTFChars(env,javaString, nativeString).} 您需要使用界面指针env来操作Java对象。总结在您的程序中使用JNI并不是一件容易的事情，然而，JNI的性能和使用原有代码的能力将会为您的Java程序添加更多的功能并且能胜任更多的挑战，如果需要关于JNI的更多信息，可以访问JNI的主页。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com