Java本地方法在Windows注册表访问中的应用

朱朝霞[1],王 杨[2]
(1.长江大学 计算机科学学院,湖北 荆州 434023;
2.苏州大学 计算机科学与技术学院,江苏 苏州 215006)

摘 要:注册表是Windows系统的核心数据库,一旦操作失败会对系统和应用程序造成不可预见的影响。作为Windows的特定属性,通过Java语言直接访问Windows注册表难以实现。首先介绍了JNI技术和实现方法,然后讨论了JNI在访问Windows注册表中的应用。通过Java本地方法调用Windows API注册表函数访问和修改注册表,有效地解决了Java应用程序访问注册表的难题,最后结合具体应用给出了实现的过程。

关键词:注册表;Java;JNI;Windows API;异常处理
中图分类号:TP316 **文献标识码**:A **文章编号**:1673-629X(2006)11-0225-03

Application about JNI How to Access to Windows Registry Table

ZHU Zhao-xia[1], WANG Yang[2]
(1. Department of Computer Science, Yangtze University, Jingzhou 434023, China;
2. School of Computer Science & Technology, Soochow University, Suzhou 215006, China)

Abstract: Registry table is the core database of Windows system, if fail to operate it, an unpredictable influence will be produced as far as system and application programmer. But as Windows' special property, direct access is very difficult to realize. Firstly introduce JNI and how to apply it to Windows' registry table. By means of using JNI and calling Windows API functions, solves the problem about Java applications how to access to registry table. And the major source codes are presented.

Key words: registry table; Java; JNI; Windows API; exception handling

0 引 言

Java语言作为一种当前盛行的网络编程语言,具有跨平台性、安全性、简单易用、类库丰富等特点。但是有些情况下,如希望直接利用C/C++程序代码、访问OS的专有特性、实时性要求较高等,借助于Java本地方法(JNI)是一种极为有效的途径[1]。

注册表是Windows用来保存操作系统及各种应用程序相关信息的文件,包含硬件、软件以及操作系统配置信息,是Windows系统和应用程序数据的核心数据库。现有的修改Windows注册表的途径有很多,如Windows自带的regedit.exe文件、超级兔子等。另外利用编程软件也能方便地修改注册表,如Visual Basic, Visual C++, Dephi等。作为Windows的特定属性,Windows注册表的程序天生就是无法移植的。标准的Java类库不支持Windows注册表的相关操作,通过Java语言直接访问Windows注册表难以实现。文中通过对JDK中的Java本地方法的分析与研究,给出了如何通过JNI(Java Native Interface)来获得对Windows注册表的修改及访问权,并编写了一个实用的注册表修改器。

收稿日期:2005-12-21
基金项目:安徽省教育厅基金资助项目(JG05003);安徽师范大学校青年基金资助项目(2005xqn05)
作者简介:朱朝霞(1973-),女,浙江杭州人,讲师,硕士,从事计算机操作系统的研究工作。

1 Java本地方法

1.1 JNI简介

JNI是Java的一种native(本地的)编程接口技术[2],由SUN公司开发,是Java平台的一个强有力的特性。JNI定义了一个标准的命名和调用准则,从而允许运行在Java虚拟机上的Java代码操作其他语言(例如C、C++、汇编语言)编写的应用程序和库。使用它提供的方法,Java代码能够直接与特定操作系统和硬件平台中的本地共享二进制库进行交互。对于Java和本地应用程序,JNI起到了纽带的作用,如图1所示。一方面JNI可使运行在

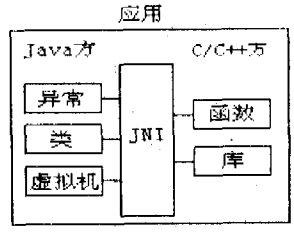

图1 JNI的纽带作用

Java 虚拟机中的 Java 代码调用 native 库函数或其他程序。另一方面，利用 Invocation API，可将 Java 虚拟机嵌入 native 应用程序中。

1.2 JNI 的实现方法

在本地方法中支持两种代码的相互调用，native 共享库及 native 应用[3]。前一种基本思想是将 C、C++ 或汇编子程序以动态链接库形式供 Java 程序调用。而后一种的核心思想是将 C/C++ 或汇编子程序先通过加载 Java 虚拟机的动态链接库来实例化虚拟机，然后将待执行的 Java 类及所需参数传给虚拟机，最后由 Java 虚拟机通过调用平台或其它动态链接库来解释执行该类。在本例中采用 native 共享库本地代码的调用。Native 共享库(例如从 Java 程序调用 C/C++ 代码)遵循以下 6 个基本步骤。

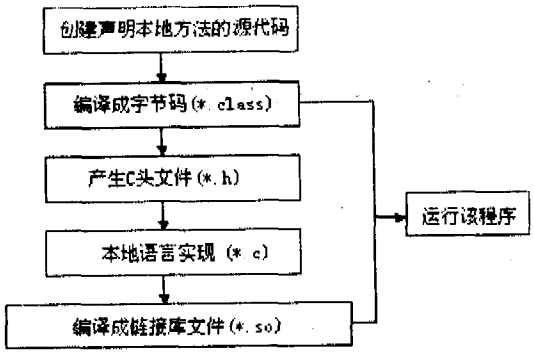

图 2 从 Java 程序调用 C/C++ 代码

在编写声明本地方法的 Java 源代码(.java)时，代码中有两个关键部分的设置：

(1)用 native 声明本地方法：native 关键字表示该方法不是一个常规方法，而是一个本地方法，该方法的具体实现不是用 Java 而是在别的文件中用其他语言完成，如 C/C++ 等，但在程序中必须包含第三步产生的 JNI 头文件。

(2)使用函数 loadLibrary 静态加载所要使用的 native 方法的共享链接库，使 Java 虚拟机能够链接到相应的本地方法。

在编写 native 方法前，需通过头文件生成器 javah 将.class 类文件生成一个 JNI 类型的头文件，为 native 方法的实现提供一个函数署名，此类头文件中包含一个带布局的结构定义，在本地方法的实现中将得到引用。

2 利用 JNI 访问 Windows 注册表

2.1 编程思想

实现一个简单接口，以便用 Java 代码来访问注册表，然后用本机代码来实现该接口。在接口中主要通过 Windows 中提供的注册表 API 访问函数来完成对注册表的修改。API(Application Programing Interface)是 Windows 提供的一个 32 位环境下的应用程序编程接口，由一组.dll 动态链接库文件组成，在程序运行时加载，为 Windows 操作系统中运行的程序提供必要的计算机服务。

Windows API 中可用于直接操作注册表的函数共有 20 多个[4,5]，根据其功能不同可以分为键管理类、值管理类、查询计数类、备份/恢复类、实用类、安全类等。其中较为常用的有：RegOpenKeyEr 打开指定键，获得其句柄；RegQuerryValueEx 获取指定值名字的类型和数据；RegCreateKeyEr 创建指定键，若已存在，则打开它；RegDeleteKeyEr 删除子键及其后代；RegCloseKey 释放前面打开关键字的句柄。

在 JNI 中利用 Windows 提供的 API 函数可方便地读取和操作注册表，来满足人们对注册表进行操作的特殊需要。

2.2 接口功能设计

在本例中主要功能设计为：枚举存放在注册表项中的所有名字；读取用某个名字存放的值；设置用某个名字存放的值。

2.3 本地方法设计

本例中，通过用 C 语言实现的本地方法(getValue, setValue, hasMoreElement, nextElement)调用 Windows API 函数来操作注册表。通过 API 函数访问注册表通常分 3 个步骤进行：第一步，建立一个新键或者打开一个已有的键，得到这个键的句柄；第二步，借助于第一步获得的句柄，使用相应的函数对该键及其键值项进行各种访问操作；第三步，释放第一步得到的句柄。

部分程序段如下：

(1)确定建立或设置的键值在注册表结构中的位置及注册表项。设置 Java 安装目录在注册表中的位置是 HKEY_LOCAL_MACHINE \ SOFTWARE \ JavaSoft。

```
public static void main(String[] args)
{
Win32RegKeykey = new
Win32RegKey(Win32RegKey.HKEY_LOCAL_MACHINE,
"SOFTWARE \ \ JavaSoft \ \ Java Development Kit \ \ 1.3.1_
01");
        key.setValue("Postgraduate","zhu zhaoxia");
        key.setValue("Specialty","Technology for Computer Ap-
plication");
        key.setValue("Lucky number",new Integer(12));
        key.setValue("odd number",new byte[]{5, 9,13,17,
21});
        ……
```

(2)声明一些变量和函数，包括注册表根键的常量和注册表操作方法等。如：

```
public static final int HKEY_CLASSES_ROOT=0x80000000;
public static final int HKEY_CURRENT_USER=0x80000001;
public static final int HKEY_LOCAL_MACHINE=0x80000002;
public static final int HKEY_USERS=0x80000003;
public static final int HKEY_CURRENT_CONFIG=0x80000005;
public static final int HKEY_DYN_DATA=0x80000006;
……
```

2.4 错误处理

为保持 Java 平台的正确运行，本地方法的设计者必

须预测错误产生的条件和处理所产生的异常,当本地方法诊断出一个它无法解决的问题时,它应该将问题报告给 Java 虚拟机,产生一个异常,但 C 语言没有异常,此时须调用 Throw 或者 ThrowNew 函数来建立新的异常对象。在本例中,笔者从打开注册表项、调用 API 函数、注册表值项的类型等方面入手,对程序中可能出错的情况进行估计,并根据错误情况抛出异常。

例如:如果打开注册表项错误,则抛出以下异常:

```
if (RegOpenKeyEx(root, cpath, 0,
KEY_READ, &hkey)! =ERROR_SUCCESS)
    {
      (*env)->ThrowNew(env,(*env)->FindClass(env,
"Win32RegKeyException"),"Open key failed");
      (*env)->ReleaseStringUTFChars(env,path,cpath);
      return NULL;
    }
```

在本例中对出现的异常问题以统一的方式进行处理,不仅增加了程序的稳定性和可读性,而且规范了程序的设计风格,有利于保证程序的质量。

2.5　程序的运行

针对以上程序在 DOS 提示符依次运行如下命令:

```
javac Win32RegKey.java
javah Win32RegKey
javah WinRegKeyNameEnumeration
cl -I d:\jdk1.3.1_01\include -I d:\jdk1.3.1_01\include
\win32 -LD Win32RegKey.c advapi32.lib -FeWin32RegKey.
dll
javac Win32RegKeyTest.java
java Win32RegKeyTest
```

如运行成功在 Windows 的 DOS 命令提示符下运行结果如图 3 所示。

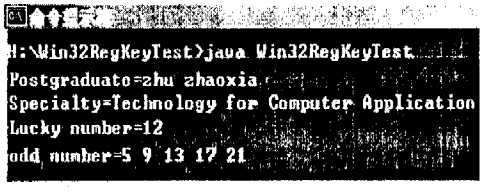

图 3　DOS 下显示的注册表中的部分键值

此时通过 Windows 注册表编辑器可观察到图 4。

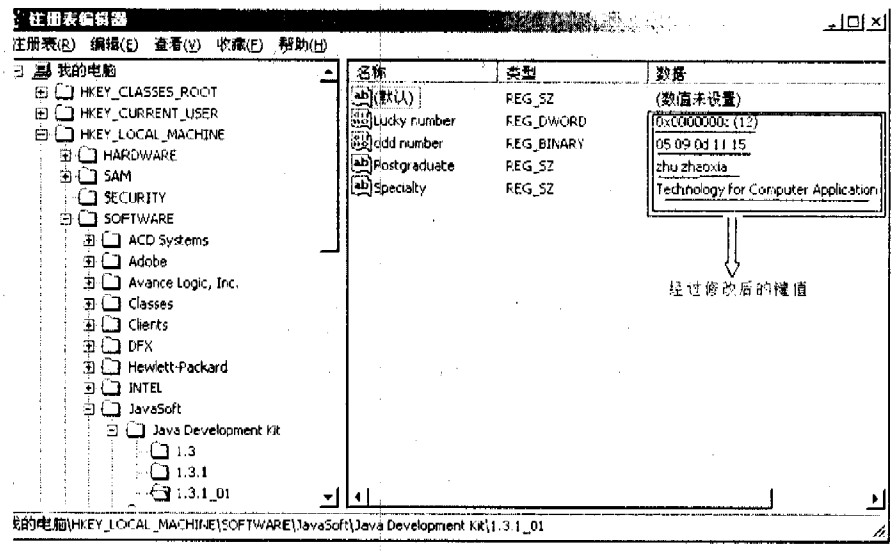

图 4　修改后的注册表内容

3　结束语

通过 SUN 公司提供的 Java 本地方法不仅解决了 Java 不能访问 Windows 注册表的难题,同时通过此实例展示了如何用 Java 平台包装程序来包装普通 C 语言的 API 子集。通过 Java 本地方法,可方便地查询及修改注册表的任何位置的有关信息。由于注册表是 Windows 的核心数据库,一旦操作失败会对系统和应用程序造成不可预见的影响,所以在操作之前一定要注意对注册表的保护工作,以免一次误操作导致系统的崩溃。目前在从事网格平台设计、分布式 DDOS 攻击的实时 IDS 研究和基于移动代理的 P2P 计算等项目过程中都需要利用 JNI 本地方法来进一步完成相关任务。

文中程序运行平台为 Windows NT 及在 JDK1.3.1_0 环境下。

参考文献:

[1] 刘晓华.精通 Java 核心技术[M]:北京:电子工业出版社,2003.

[2] Horstmann C S. 最新 Java 2 核心技术[M]:北京:机械工业出版社,2003.

[3] 管贻生. Java 高级实用编程[M]:北京:清华大学出版社,2004.

[4] SUN Corp. Java Application WITH Java Native Interface[EB/OL]. 1994. http://java.sun.com/docs/books/tutorial/native.1/2002/2004.

[5] LIRON T. Enhance Your Java Application with Java Nation Interface[EB/OL]. 1998. http://www.public.asu.edu/~wjanjua/java/jni/2003/2004.

(上接第 224 页)

7(1):118-127.

[10] Deng Jing, Varshney P K, Haas Z J. A New Backoff Algorithm for the IEEE 802.11 Distributed Coordination Function [EB/OL]. 2003-10. http://www.ees.syr.edu/research/SensorFusionLab/Downloads/Jing%20Deng/lmild_cnds04.pdf.