Java ME 蓝牙串口通讯中异地文件目录结构的动态访问控制①

梁旗军[1]　吴喜兰[2]　罗海平[1]　(1.南昌航空大学 计算机学院 江西 南昌 330063;
2.江西科技师范学院 江西 南昌 330016)

摘　要： 在多终端的 Java ME 蓝牙串口通讯中，经常需要访问并控制异地文件目录结构。为了保证在串口连接的条件下能够便捷和实时的获取目录结构信息，准确定位文件，还必须对目录结构动态的进行控制。针对上述问题，研究并给出了适用于动态访问控制的二维向量存储结构及访问控制方法的设计，同时提供了该方法应用在蓝牙串口通讯中的实现程序。实际应用结果也表明了动态访问控制的有效性。

关键词： Java ME；蓝牙；串口通讯；二维向量；目录结构；动态访问

Dynamic Access Control of Remote File Directory Structure in Java ME Bluetooth Serial Communication

LIANG Qi-Jun[1], WU Xi-Lan[2], LUO Hai-Ping[1]
(1.Computer School, Nanchang Hangkong University, Nanchang 330063, China;
2.Jiangxi Science&Technology Normal University, Nanchang 330016, China)

Abstract: In multi-terminal Java ME Bluetooth serial port communications, it is necessary to access and control remote file directory structure. To ensure that under the conditions of serial connections it is convenient and real-time to access the directory structure information, and locate the file accurately, the directory structure must also be controlled dynamically. To address these problems, this paper researches and designs the two-dimensional vector storage structure and access control methods, along with the application and the realization of the method in the Bluetooth serial port communication. The results of practical application show the effectiveness of dynamic access control.

Keywords: Java ME; Bluetooth; serial port communication; two-dimensional vector; directory structure; dynamic access

1　引言

Java ME 为运行在移动和嵌入式设备上的程序提供了一个健壮、灵活的环境，如：智能手机、PDA、电视机顶盒和打印机等。其前身即 J2ME[1,2]。在蓝牙通讯中，各种移动和嵌入式设备可以利用已有的或自建的协议建立并保持通信。而在实际的应用开发中，出于简单便捷性和实时性的要求，通常会采用基于 RFCOMM 协议的串口通讯技术。在某些应用中会要求动态访问控制异地的文件目录结构，如访问或控制网络中另一终端某种类型的某个文件等。但是基于 RFCOMM 的通讯有数据量和格式的限制，从提高实时性的方面讲，每次传输的数据量越小越好，时间越短越好。因此，如何在 Java ME 蓝牙串口通讯中高效的动态对异地的文件目录结构进行访问和控制存储操

① 收稿时间:2009-05-21

作，也就成为基于 RFCOMM 应用开发的一个重要工作。

针对上述问题，本文研究了二维向量存储结构，重点探讨了结合二维向量的异地文件目录的动态访问控制方法的设计，并给出了在 Java ME 蓝牙串口通讯中的实现程序。

2 二维向量存储结构

各个终端访问控制异地终端的文件目录结构，不仅要保存当前文件的当前路径的目录结构，同时要保存要访问的文件的名称。由于当前路径的深度和名称可能不尽相同，故要求实现目录结构存储的存储结构要能够处理名称不定长的情况。同时为了便于查找当前路径上的某个目录，要求存储结构具有向下的链表的功能。除此之外，当需要回退访问已访问过的目录或文件时，又要求存储结构具有向上的链表的功能。

Java ME 平台的支撑语言是 java。在 java 语言中可以满足处理不定长及链表要求的最常用的是 Vector 向量类。Vector 向类实现了一个可增长的数组对象。象数组一样，它包含了可用整数索引访问的组件。Vector 的大小可以根据需要增缩。Vector 向量类与 ArrayList 相似，都扩展了 AbstractList。它可以像聚集对象一样工作，用来实现列表。并且所有访问 Vector 内容的方法都是同步的[3]。

但是由于 Vector 向量类中的元素的访问依靠整数索引，只通过一维向量实现回退访问已访问过的目录或文件很困难，在这里采用二维向量的方式加以解决。通过二维向量构成一个逻辑矩阵，其横向量存储了当前目录下所有的文件夹和经过过滤的某种固定后缀的文件，纵向量存储了用户访问文件的记录。如图 1 所示。

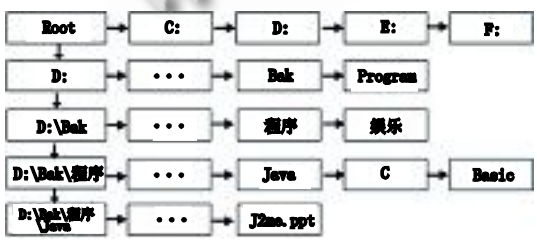

图 1 二维向量逻辑矩阵

图 1 给出了异地终端中目录结构为 D:\Bak\程序\Java\J2me.ppt 的 PPT 文件在本地终端中的存储情况。从图中可以看到名称为 J2me.ppt 的元素，其所在的横向量保存了其所在目录中的所有文件夹和文件名称，纵向量为上一级已访问的目录名称。

3 动态访问控制方法设计

异地文件目录结构的动态访问控制主要包括存储和查找两大操作。其中存储是指将异地文件所在的目录名称和文件名保存在二维向量中，查找是指通过控制二维向量存储结构中的向量及向量元素，完成三个主要任务：开启系统根目录、开启上级目录和开启下级目录。

为了减少 RFCOMM 通讯的数据量和简化数据格式，本方法提出的动态访问控制的设计思想是：本地终端和异地终端可互为服务器和客户机，异地终端上建立二维向量并保存目录结构，同时根据接收的控制命令标识完成访问控制操作，形成目录和文件列表，发送到本地终端；本地终端动态发布控制命令，通过命令标识的形式发送到异地终端，同时接收异地终端传输的目录和文件列表。

3.1 二维向量存储结构的约定

要通过二维向量存储异地目录结构，首先必须对向量元素数据类型及特殊标识作以下几个约定：

① 存储元素为不定长字符串。

② 横向量存储了当前目录下所有的文件夹和经过过滤的某种固定后缀的文件。

③ 纵向量存储了用户访问文件的记录。

④ “Root”字符串作为根目录的标识符。

⑤ “…”字符串作为向上翻页的标识符。

3.2 获取目录结构

进行异地文件目录结构的动态操作的首要条件就是获取目录结构。因为各个终端都是以 java 语言为支撑，在文件操作部分，java 语言提供了一个 File 类，它实现了 Serializable 和 Comparable 两个接口，是文件和目录名的抽象表示方法。

File 类提供了几个返回目录下文件名和目录名数组及测试目录的成员函数[4]。这几个函数定义如下：

① static File[] listRoots()

② boolean isDirectory()

③ String[] list()

其中，①返回文件系统的根目录下文件列表，可用于构建存储根目录首个向量；②测试是否是一个目

录，可用于开启上级目录和开启下级目录中当做判断条件；③返回该目录下的文件名和目录名数组，可用于构建存储结构中的横向量。

3.3 动态访问控制分析

将整个异地文件目录结构的动态访问控制作为一个系统，运用面向对象的系统建模与设计方法进行分析[5]，不难得到整个系统中的参与者或角色有三个，分别为：本地终端、异地终端和用户。具体用例包括：建立蓝牙串口通讯、构建二维向量存储结构、动态访问控制等。据此可以得到系统用例图如图 2 所示：

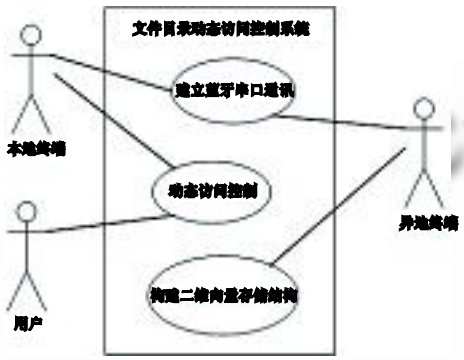

图 2　系统用例图

因为系统流程并不复杂，因此可以将系统的所有用例和整个系统流程结合起来考虑，用一张系统顺序图表示，如图 3 所示。

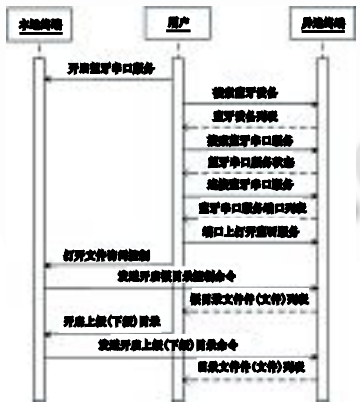

图 3　系统顺序图

从图 3 显示的消息的顺序，可以看出整个系统应用是由用户发起的，在本地终端和异地终端之间建立了蓝牙串口通讯，由本地终端发出打开文件访问控制、开启根目录、上级和下级目录命令，异地终端处理后，返回目录文件夹(文件)列表给用户，在本地终端显示这个列表。

4　蓝牙串口通讯中实现动态访问控制

在 Java ME 中，提供了 RFCOMM 协议完成串口通讯[6]。RFCOMM 是蓝牙软件协议栈的一部分，该层位于 L2CAP 之上，基于 ETSI 标准 TS07.10 仿真 9 针 RS232 串口的功能，提供了模拟标准串口通信的能力[7]。因此，要实现 Java ME 蓝牙串口通讯中的异地文件目录结构的动态访问控制，首先要建立 RFCOMM 连接。在串口连接的基础上，进行开启根目录、上级和下级目录的访问控制操作。

4.1 蓝牙串口通讯的连接及数据传输

在蓝牙应用服务中，每一个蓝牙应用都表现为一个蓝牙服务，提供服务的为服务器，消费服务的为客户端。无论客户端还是服务器端首先都要进行初始化，服务器端再创建服务、注册服务、等待用户访问、创建连接提供客户端消费，而客户端要发现周围服务和设备、消费服务[1]。

现在以服务器端为例，给出启动服务的实现代码，可以用一个线程实现，包括搜索本地设备，开启 RFCOMM 协议服务，等待连接等。具体实现代码如下：

```
//启动服务
public void run()
{isRunning = true;
try{
LocalDevice local =
LocalDevice.getLocalDevice();//本地蓝牙设备对象;
local.setDiscoverable(DiscoveryAgent.GIAC;
//将本地设为通用查找模式
}catch (BluetoothStateException e){.....}
Try
{server = (StreamConnectionNotifier)
Connector.open("btspp://localhost:111111
11111111111111111111111111");
//通过128位的 RFCOMM协议开启服务
}catch (IOException e){.....  }
while (isRunning)
{StreamConnection conn = null;// 定义一个
RFCOMM连接对象
try{try{conn = server.acceptAndOpen();//等待客
户连接}
```

```
catch (IOException e){return;}
dis = conn.openDataInputStream();//获得输入流
dos = conn.openDataOutputStream();//获得输出流
while (isRunning)//循环监听输入流
{if (dis.available() > 0)//判断输入流中是否有信息
   {byte[] buf = new byte[100];
   this.dis.read(buf);
   String str = new String(buf, "GBK").
   trim();//获取信息
   dataFormat(str);//格式化信息}
   Thread.sleep(200);}}
   catch (IOException e){.....}
   catch (InterruptedException e){.....}
   finally {if (conn != null)
   {try    {conn.close();
   conn = null;}
   catch (Exception e) {e.printStackTrace ();}}}}
```

蓝牙串口通讯连接建立之后，要进行异地终端和本地终端之间的数据传输，传输的内容是目录及文件信息，即是二维向量中的某个横向量的内容。

首先定义一个全局变量 vecDirs，用于存放目录结构，它是一个二维向量；定义如下：

```
Vector<Vector> vecDirs;//存放目录结构的二维向量
```

根据约定横向量的元素是不定长字符串，所以可以很容易将其转化为字节形式，利用流来进行传输。具体实现见 send()方法，代码如下：

```
//发送目录及文件信息
private void send()
{for (int i = 1; i < vecDirs. lastElement(). Size (); i++)
{try  {Thread.sleep(300);
outputStream.write((strOpenDir +
hundred(i) +
vecDirs.lastElement().get(i)).getBytes());
outputStream.flush();}
catch (IOException e)
{e.printStackTrace();}
catch (InterruptedException e)
{e.printStackTrace();}
}}
```

4.2 开启系统根目录和开启上级目录

由于 vecDirs 中的元素是向量，可以再在具体方法中再次定义元素为 String 的向量 vecFiles，用于存放当前目录下文件夹和文件。定义如下：

```
Vector<String> vecFiles;//存放当前目录下文件夹和文件的向量
```

通过首个向量来存储根目录，首先添加“Root”字符串作为根目录的标识符，然后遍历系统根目录下的列表，逐一添加进向量 vecFiles。最后将根向量添加至用户访问目录记录的存储向量 vecDirs 中。实现代码如下：

```
//开启系统根目录
private void openRoot()
{Vector<String> vecFiles = new
Vector<String>();
vecFiles.add("Root");
File[] roots = File.listRoots();
for (int i = 0; i < roots.length; i++)
{vecFiles.add(roots[i].toString());}
vecDirs.add(vecFiles);
send();   }
```

从图 1 的二维向量逻辑矩阵中可以看出，要想返回到上一层目录中，只需要删除用户最后一次的文件访问记录就可以，即去除二维向量逻辑矩阵中的最后一个向量。再将上一层目录的所有信息发送给终端显示。实现代码如下：

```
//开启上层目录
private void openUpDir()
{vecDirs.remove(vecDirs.size() - 1);
send();}
```

4.3 开启下级目录

开启下级目录是二维向量动态访问控制方法的关键，在这里向上开启做为向下开启的一个子方法进行定义。同时设置一个标识 index，其值为 1 时表示向上开启。

开启下级目录首先判断当前目录是否是根目录，如若是根目录，则直接进行下一级目录的遍历和文件格式的筛选，否则再次进行筛选。再次进行筛选的过程先判断 index 的值，如为 1，说明用户采取的是向上开启，调用 openUpDir()函数；否则继续往下进行目录的遍历和文件的筛选。整个过程直至获得了用户

指定的的文件结束。过程流程图如图 4 所示。

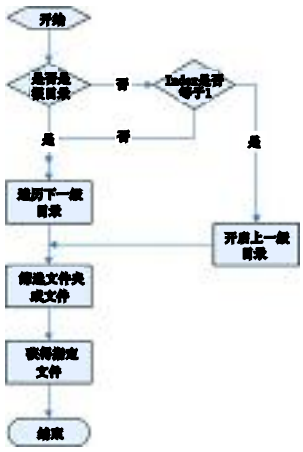

图 4　开启下级目录流程图

根据图 4，以筛选 PPT 文件为例，编写实现代码如下：

```
//开启下层目录
private void openDownDir(int index)
{Vector<String> vecFiles = new
Vector<String>();
String str =
vecDirs.lastElement().get(index).toString();  if
(vecDirs.lastElement().get(0).equals("Root")
{vecFiles.add(str); vecFiles.add("...");
File file = new File(str);
String[] filelist = file.list();
for (int i = 0; i < filelist.length; i++)
{if (new File(str + filelist[i]).isDirectory())//筛选文件夹
{vecFiles.add(filelist[i]);}
//筛选 ppt 文件
if(filelist[i].endsWith(".ppt")||
filelist[i].endsWith(".PPT"))
{vecFiles.add(filelist[i]);}}
vecDirs.add(vecFiles);
send();}
else
{if (index == 1){openUpDir();}
else{String strDir
vecDirs.lastElement().get(0).toString() + str;
File file = new File(strDir);
if (file.isDirectory())//是文件夹
{vecFiles.add(strDir + "\\");
vecFiles.add("...");
String[] filelist = file.list();
for (int i = 0; i < filelist.length; i++)
{if (new File(strDir + "\\" +
filelist[i]).isDirectory())
{vecFiles.add(filelist[i]);}
if (filelist[i].endsWith(".ppt")||
filelist[i].endsWith(".PPT"))
{vecFiles.add(filelist[i]);}}
vecDirs.add(vecFiles);
send();}
else//是 ppt 文件
{openPPT(strDir);}}}}
```

5　结语

在串口通讯中对异地文件目录结构的动态访问控制，是以串口连接的多终端应用中的常常需要的重要功能。通过在笔者开发的智能手机教辅工具软件开发项目中应用本文提出的二维向量的存储结构，证明了该方法的有效性。未来的研究工作将进一步考虑数据传输的安全性问题，以达到更好的控制效果。

参考文献

1 詹建飞.J2ME 开发精解.北京:电子工业出版社, 2006.
2 Jode MD. Symbian OS J2ME 编程指南.北京:人民邮电出版社, 2005.
3 许满武.Java 程序设计.北京:高等教育出版社, 2006.392.
4 张曜,郭立山,游泳明.Java 函数实用手册.北京:冶金工业出版社, 2003.50,144－145.
5 车皓阳,杨眉.UML 面向对象建模与设计(第 2 版).北京:人民邮电出版社, 2006.106－111.
6 André N. Klingsheim. J2ME Bluetooth Programming. Department of Informatics University of Bergen. Master's Thesis. 2004.52－53.
7 Borja Gomez Zarceño. Mobile Applications with J2ME and JSR82: Bluetooth enabled Java Applications for Mobile Phones. ACS seminar. 2004.