串行通讯之
UARTLoopback

Hanford

2016 年 11 月 18 日

目 录

第 1 章　串行通讯之 UARTLoopback

1　USB 转串口

这两天在做 Android 手机上的串行通讯程序。手机没有串口，所以使用了 USB 转串口，如下图所示：

图 1　USB 转串口

上图中，红色的 USB A 型插头用来给此设备供电；黑色的 Micro USB 插头用来连接 Android 手机；粉红色的 9 针插头用来连接串口设备。

购买此产品时，附带了 Java 源代码，也就是工程 UARTLoopback。本文对其进行说明及改进。

2　USB Accessory

USB 设备分为两大类：USB Host、USB Accessory（USB 附件）。USB 键盘、鼠标连入手机后，由手机给其供电，它们属于 USB Host；上面的 USB 转串口连入手机后，会给自己、手机供电，它属于 USB Accessory。

查看 UARTLoopback 的代码可知：访问 USB 转串口的实质是访问 USB Accessory。

关于 USB Accessory 的更多信息请参考如下博客：

http://blog.csdn.net/yingzhao80/article/details/45511351

3　连入手机

Android 手机上安装 UARTLoopbackActivity.apk 后，将 USB 转串口接入手机，就会弹出如下界面：

图 2

这是如何实现的呢？请查看 UARTLoopback 的 AndroidManifest.xml 文件。下面是精简后的内容，重点是红色字体部分：

```
...     ...     ...
<uses-feature  android:name="android.hardware.usb.accessory"/>
...     ...     ...
<intent-filter>
        <action  android:name="android.hardware.usb.action.USB_ACCESSORY
_ATTACHED"/>
</intent-filter>
<meta-data
        android:name="android.hardware.usb.action.USB_ACCESSORY_ATTAC
HED"
        android:resource="@xml/accessory_filter">
</meta-data>
...     ...     ...
```

4　代码改进

串行通讯的核心类就是 FT311UARTInterface，笔者对其进行了改进。改进版的下载网址为：http://download.csdn.net/detail/hanford/9686781

下文的说明以改进版为准。

5　打开串口

打开串口的代码如下

```
com.UARTLoopback.FT311UARTInterface m_Comm = new com.UARTLoopback.FT311UARTInterface(this);
if(m_Comm.open(9600,'N',8,1,0))
{//成功打开串口
}
else  if(m_Comm.isExist())
{//打开串口失败，可能是权限不够，申请权限
      m_Comm.requestPermission();
}
else
{//说明手机未连接 USB 转串口
}
```

m_Comm.open 用来打开串口

m_Comm.isExist 用来判断 USB 转串口是否已经插入手机

m_Comm.requestPermission 用来申请权限

打开串口的时候就设置通讯参数，为什么这么设计呢？因为：从 USB 转串口插入手机到拔出手机这段时间内，只能配置一次通讯参数。

6　写串口数据

请参考下面的代码

```
if(m_Comm.isOpen()) {
      byte[] data = m_txtSend.getText().toString().getBytes();
      m_Comm.write(data,data.length);
}
```

m_Comm.isOpen 判断串口是否已经打开

m_txtSend.getText().toString().getBytes() 获取文本框 m_txtSend 内的文本，然后转换为二进制数据

m_Comm.write 发送二进制数据

7　主动读取串口数据

请参考下面的代码

```
if(m_Comm.isOpen()) {
    byte[] data = new byte[1024];
    int nRead=m_Comm.read(data, data.length);
    try {
        m_txtRecv.setText(new String(data, 0, nRead, "UTF-8"));
    } catch (UnsupportedEncodingException ex) {
    }
}
```

m_Comm.read 用来读取串口数据，返回读取到的字节数。接下来的代码，将读取到的二进制数据转换为字符串，并显示到文本框 m_txtRecv 里。

8　被动读取串口数据

被动读取串口数据，就是一旦获得了串口数据就通知程序。其代码有点多：

```
m_Comm.setEventDataReceived(m_EventDataReceived);
com.UARTLoopback.FT311UARTInterface.EventDataReceived m_EventData
Received = new com.UARTLoopback.FT311UARTInterface.EventDataRecei
ved(){
    public void onEvent(byte[] data,int nBytes)
    {//接收到串口数据，就调用此函数
        try {
            m_sRecv += new String(data, 0, nBytes, "UTF-8");
        } catch (UnsupportedEncodingException ex) {
        }
        m_Handler.sendEmptyMessage(1);  //更新界面显示
    }
};
private Handler m_Handler = new Handler() {
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case 1: m_txtRecv.setText(m_sRecv); break;
        }
```

```
            super.handleMessage(msg);
        }
};
```

代码 m_Comm.setEventDataReceived(m_EventDataReceived);表示一旦接收到串口数据，马上调用 m_EventDataReceived 对象的 onEvent 函数。

onEvent 函数中，将串口数据（保存在数组 byte[] data 里，字节数为 nBytes）转换为文本，然后加到字符串变量 m_sRecv 的右边。

因为 onEvent 函数不在主线程里，所以需要代码 m_Handler.sendEmptyMessage(1);通知 m_Handler 更新主界面。其实就是 handleMessage 函数中的 m_txtRecv.setText(m_sRecv)被执行。

总结：

1）m_Comm.setEventDataReceived 指定事件处理对象，一旦读取到串口数据，将调用该对象的 onEvent 函数；

2）onEvent 函数是被多线程调用的，更新主界面请使用 Handler、sendEmptyMessage；

3）如果 m_Comm.setEventDataReceived 的参数不是 null，那么就无法主动读取串口数据了。也就是说，此时 m_Comm.read 始终返回 0。

9　关闭串口

关闭串口的代码很简单，如下所示：

```
m_Comm.close();
```

不过，它的问题最严重：

调用上述代码，读取串口数据的线程（FT311UARTInterface.ThreadRead.run）将被阻塞在如下代码行：

```
nRead    =    FileInputStream_read(m_InputStream,data,data.length);
```

上面的代码调用了 FileInputStream.read 函数，这是一个同步函数——没有读取到串口数据，就不会返回。这个时候，如果串口设备发送过来数据，线程将正常退出；如果串口设备一直未发送数据过来，那么这个线程将永远阻塞在这一行上。

线程 ThreadRead 阻塞后，m_Comm.open 将无法再打开串口。解决办法就是：拔下 USB 转串口，重新插入。

总结：关闭串口会极大概率的导致一个僵尸线程的产生，不够完美的解决办法就是重新拔、插 USB 转串口。