首先放一下我这次使用的工具

链接: https://pan.baidu.com/s/1o78lHX4 密码: vg7w

com0com0是windows内核模式下的虚拟串口驱动程序。可以创建虚拟串口对来连接两个COM端口

在这里简单的介绍一下com0com0的使用方法

首先，下载com0com-3.0.0.0-i386-and-x64-unsigned，解压缩并安装



接下来我们就可以使用setup command prompt来进行模拟串口对了

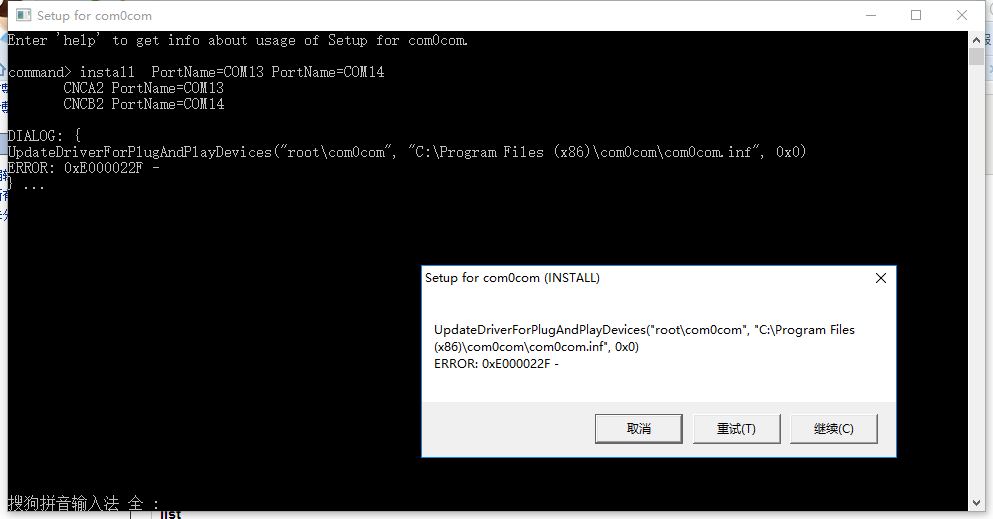
在这里首先介绍两个命令

**install  PortName=串口名1 PortName=串口名2**

**新建两个虚拟串口，并连接为串口对**

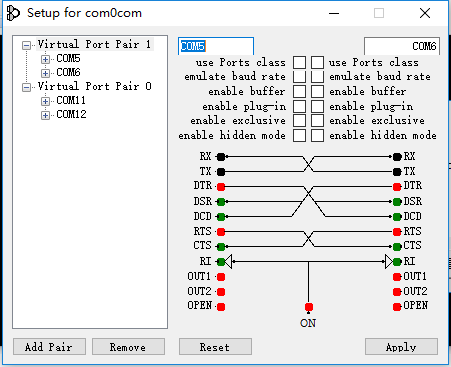
**list**

**列出目前所有的虚拟串口**

****

如果跳出更新驱动的提示可以暂时无视，不影响后续的操作。

另一个工具setup具有图形化界面，可以查看目前的串口对



接下来介绍java编程的时候需要用到的两个jar包

 RXTXcomm.jar

serialPortIO.jar

同时，在使用这两个jar包进行java编程之前，我们要将mfz-rxtx-2.2-20081207-win-x64压缩包下的两个dll文件放到C:\Windows\System32下

然后，这里是一个简单的串口读写的例子

import gnu.io.SerialPort;

import serialPort.SerialTool;

public class Read {

public static void main(String[] args) throws Exception{

// TODO Auto-generated method stub

//以波特率115200打开串口COM12

SerialPort serialPort = SerialTool.openPort("COM12", 115200);

SerialTool.addListener(serialPort, new SerialListener(serialPort));

}

}

import java.util.Scanner;

import gnu.io.SerialPort;

import serialPort.SerialTool;

public class Write {

public static void main(String[] args) throws Exception {

Scanner scan = new Scanner(System.in);

String temp;

//以波特率115200打开串口COM11

SerialPort serialPort = SerialTool.openPort("COM11", 115200);

while(scan.hasNext())

{

temp=scan.nextLine();

SerialTool.sendToPort(serialPort, temp.getBytes());

}

}

}

import javax.swing.JOptionPane;

import gnu.io.SerialPort;

import gnu.io.SerialPortEvent;

import gnu.io.SerialPortEventListener;

import serialPort.SerialTool;

public class SerialListener implements SerialPortEventListener {

public SerialPort serialPort;

public SerialListener(SerialPort serialPort)

{

this.serialPort=serialPort;

}

public void serialEvent(SerialPortEvent serialPortEvent) {

switch (serialPortEvent.getEventType()) {

case SerialPortEvent.BI: // 10 通讯中断

System.out.println("与串口设备通讯中断");

break;

case SerialPortEvent.OE: // 7 溢位（溢出）错误

case SerialPortEvent.FE: // 9 帧错误

case SerialPortEvent.PE: // 8 奇偶校验错误

case SerialPortEvent.CD: // 6 载波检测

case SerialPortEvent.CTS: // 3 清除待发送数据

case SerialPortEvent.DSR: // 4 待发送数据准备好了

case SerialPortEvent.RI: // 5 振铃指示

case SerialPortEvent.OUTPUT\_BUFFER\_EMPTY: // 2 输出缓冲区已清空

break;

case SerialPortEvent.DATA\_AVAILABLE: // 1 串口存在可用数据

//System.out.println("found data");

byte[] data = null;

try {

if (serialPort == null) {

System.out.println("串口对象为空！监听失败");

}

else {

data = SerialTool.readFromPort(serialPort); //读取数据，存入字节数组

//System.out.println(new String(data));

JOptionPane.showInputDialog(new String(data));

//String dataOriginal = new String(data); //将字节数组数据转换位为保存了原始数据的字符串

}

} catch (Exception e) {

System.exit(0);

}

break;

}

}

}