**Android 串口编程原理和实现方式(附源码)**

2011年11月28日 23:40:25 [tangcheng\_ok](https://me.csdn.net/tangcheng_ok" \t "_blank) 阅读数 69825更多

所属专栏： [Android应用开发总结与分享](https://blog.csdn.net/column/details/android-apps.html)

版权声明：本文为博主原创文章，遵循[CC 4.0 by-sa](http://creativecommons.org/licenses/by-sa/4.0/)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/tangcheng_ok/article/details/7021470>

    提到串口编程，就不得不提到JNI，不得不提到JavaAPI中的文件描述符类：FileDescriptor。下面我分别对JNI、FileDescriptor以及串口的一些知识点和实现的源码进行分析说明。这里主要是参考了开源项目[android-serialport-api](http://code.google.com/p/android-serialport-api/)。

    串口编程需要了解的基本知识点：对于串口编程，我们只需对串口进行一系列的设置，然后打开串口，这些操作我们可以参考[串口调试助手的源码](http://www.gjwtech.com/vcandc/scommassistantcode.htm)进行学习。在Java中如果要实现串口的读写功能只需操作文件设备类:FileDescriptor即可，其他的事都由驱动来完成不用多管！当然，你想了解，那就得看驱动代码了。这里并不打算对驱动进行说明，只初略阐述应用层的实现方式。

**（一）JNI：**

    关于JNI的文章网上有很多，不再多做解释，想详细了解的朋友可以查看[云中漫步](http://my.unix-center.net/~Simon_fu/?cat=3&paged=2)的技术文章，写得很好，分析也很全面，那么在这篇拙文中我强调3点：

    1、如何将编译好的SO文件打包到APK中？（方法很简单，直接在工程目录下新建文件夹 libs/armeabi，将SO文件Copy到此目录即可）

    2、命名要注意的地方？（在编译好的SO文件中，将文件重命名为：libfilename.so即可。其中filename.so是编译好后生成的文件）

    3、MakeFile文件的编写（不用多说，可以直接参考package/apps目录下用到JNI的相关项目写法）

    这是关键的代码：

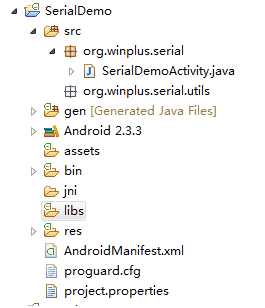
1. int fd;
2. speed\_t speed;
3. jobject mFileDescriptor;
5. */\* Check arguments \*/*
6. {
7. speed = getBaudrate(baudrate);
8. if (speed == -1) {
9. */\* TODO: throw an exception \*/*
10. LOGE("Invalid baudrate");
11. return NULL;
12. }
13. }
15. */\* Opening device \*/*
16. {
17. jboolean iscopy;
18. const char \*path\_utf = (\*env)->GetStringUTFChars(env, path, &iscopy);
19. LOGD("Opening serial port %s with flags 0x%x", path\_utf, O\_RDWR | flags);
20. fd = open(path\_utf, O\_RDWR | flags);
21. LOGD("open() fd = %d", fd);
22. (\*env)->ReleaseStringUTFChars(env, path, path\_utf);
23. if (fd == -1)
24. {
25. */\* Throw an exception \*/*
26. LOGE("Cannot open port");
27. */\* TODO: throw an exception \*/*
28. return NULL;
29. }
30. }
32. */\* Configure device \*/*
33. {
34. struct termios cfg;
35. LOGD("Configuring serial port");
36. if (tcgetattr(fd, &cfg))
37. {
38. LOGE("tcgetattr() failed");
39. close(fd);
40. */\* TODO: throw an exception \*/*
41. return NULL;
42. }
44. cfmakeraw(&cfg);
45. cfsetispeed(&cfg, speed);
46. cfsetospeed(&cfg, speed);
48. if (tcsetattr(fd, TCSANOW, &cfg))
49. {
50. LOGE("tcsetattr() failed");
51. close(fd);
52. */\* TODO: throw an exception \*/*
53. return NULL;
54. }
55. }

**（二）FileDescritor：**

    文件描述符类的实例用作与基础机器有关的某种结构的不透明句柄，该结构表示开放文件、开放套接字或者字节的另一个源或接收者。文件描述符的主要实际用途是创建一个包含该结构的FileInputStream 或FileOutputStream。这是API的描述，不太好理解，其实可简单的理解为：FileDescritor就是对一个文件进行读写。

**（三）实现串口通信细节**

1)  建工程：SerialDemo包名：org.winplus.serial，并在工程目录下新建jni和libs两个文件夹和一个org.winplus.serial.utils，如下图：



2) 新建一个类：SerialPortFinder，添加如下代码：

1. package org.winplus.serial.utils;
3. import java.io.File;
4. import java.io.FileReader;
5. import java.io.IOException;
6. import java.io.LineNumberReader;
7. import java.util.Iterator;
8. import java.util.Vector;
10. import android.util.Log;
12. public class SerialPortFinder {
14. private static final String TAG = "SerialPort";
16. private Vector<Driver> mDrivers = null;
18. public class Driver {
19. public Driver(String name, String root) {
20. mDriverName = name;
21. mDeviceRoot = root;
22. }
24. private String mDriverName;
25. private String mDeviceRoot;
26. Vector<File> mDevices = null;
28. public Vector<File> getDevices() {
29. if (mDevices == null) {
30. mDevices = new Vector<File>();
31. File dev = new File("/dev");
32. File[] files = dev.listFiles();
33. int i;
34. for (i = 0; i < files.length; i++) {
35. if (files[i].getAbsolutePath().startsWith(mDeviceRoot)) {
36. Log.d(TAG, "Found new device: " + files[i]);
37. mDevices.add(files[i]);
38. }
39. }
40. }
41. return mDevices;
42. }
44. public String getName() {
45. return mDriverName;
46. }
47. }
49. Vector<Driver> getDrivers() throws IOException {
50. if (mDrivers == null) {
51. mDrivers = new Vector<Driver>();
52. LineNumberReader r = new LineNumberReader(new FileReader(
53. "/proc/tty/drivers"));
54. String l;
55. while ((l = r.readLine()) != null) {
56. *// Issue 3:*
57. *// Since driver name may contain spaces, we do not extract*
58. *// driver name with split()*
59. String drivername = l.substring(0, 0x15).trim();
60. String[] w = l.split(" +");
61. if ((w.length >= 5) && (w[w.length - 1].equals("serial"))) {
62. Log.d(TAG, "Found new driver " + drivername + " on "
63. + w[w.length - 4]);
64. mDrivers.add(new Driver(drivername, w[w.length - 4]));
65. }
66. }
67. r.close();
68. }
69. return mDrivers;
70. }
72. public String[] getAllDevices() {
73. Vector<String> devices = new Vector<String>();
74. *// Parse each driver*
75. Iterator<Driver> itdriv;
76. try {
77. itdriv = getDrivers().iterator();
78. while (itdriv.hasNext()) {
79. Driver driver = itdriv.next();
80. Iterator<File> itdev = driver.getDevices().iterator();
81. while (itdev.hasNext()) {
82. String device = itdev.next().getName();
83. String value = String.format("%s (%s)", device,
84. driver.getName());
85. devices.add(value);
86. }
87. }
88. } catch (IOException e) {
89. e.printStackTrace();
90. }
91. return devices.toArray(new String[devices.size()]);
92. }
94. public String[] getAllDevicesPath() {
95. Vector<String> devices = new Vector<String>();
96. *// Parse each driver*
97. Iterator<Driver> itdriv;
98. try {
99. itdriv = getDrivers().iterator();
100. while (itdriv.hasNext()) {
101. Driver driver = itdriv.next();
102. Iterator<File> itdev = driver.getDevices().iterator();
103. while (itdev.hasNext()) {
104. String device = itdev.next().getAbsolutePath();
105. devices.add(device);
106. }
107. }
108. } catch (IOException e) {
109. e.printStackTrace();
110. }
111. return devices.toArray(new String[devices.size()]);
112. }
113. }

上面这个类在“[android-serialport-api串口工具测试随笔](http://blog.csdn.net/lightsoure/article/details/6833450" \o "android-serialport-api串口工具测试随笔)”中有详细的说明，我就不多说了。

3）新建SerialPort类，这个类主要用来加载SO文件，通过JNI的方式打开关闭串口

1. package org.winplus.serial.utils;
3. import java.io.File;
4. import java.io.FileDescriptor;
5. import java.io.FileInputStream;
6. import java.io.FileOutputStream;
7. import java.io.IOException;
8. import java.io.InputStream;
9. import java.io.OutputStream;
11. import android.util.Log;
13. public class SerialPort {
14. private static final String TAG = "SerialPort";
16. */\**
17. *\* Do not remove or rename the field mFd: it is used by native method*
18. *\* close();*
19. *\*/*
20. private FileDescriptor mFd;
21. private FileInputStream mFileInputStream;
22. private FileOutputStream mFileOutputStream;
24. public SerialPort(File device, int baudrate, int flags)
25. throws SecurityException, IOException {
27. */\* Check access permission \*/*
28. if (!device.canRead() || !device.canWrite()) {
29. try {
30. */\* Missing read/write permission, trying to chmod the file \*/*
31. Process su;
32. su = Runtime.getRuntime().exec("/system/bin/su");
33. String cmd = "chmod 666 " + device.getAbsolutePath() + "\n"
34. + "exit\n";
35. su.getOutputStream().write(cmd.getBytes());
36. if ((su.waitFor() != 0) || !device.canRead()
37. || !device.canWrite()) {
38. throw new SecurityException();
39. }
40. } catch (Exception e) {
41. e.printStackTrace();
42. throw new SecurityException();
43. }
44. }
46. mFd = open(device.getAbsolutePath(), baudrate, flags);
47. if (mFd == null) {
48. Log.e(TAG, "native open returns null");
49. throw new IOException();
50. }
51. mFileInputStream = new FileInputStream(mFd);
52. mFileOutputStream = new FileOutputStream(mFd);
53. }
55. *// Getters and setters*
56. public InputStream getInputStream() {
57. return mFileInputStream;
58. }
60. public OutputStream getOutputStream() {
61. return mFileOutputStream;
62. }
64. *// JNI*
65. private native static FileDescriptor open(String path, int baudrate,
66. int flags);
68. public native void close();
70. static {
71. System.loadLibrary("serial\_port");
72. }
73. }

4） 新建一个MyApplication 继承android.app.Application，用来对串口进行初始化和关闭串口

1. package org.winplus.serial;
3. import java.io.File;
4. import java.io.IOException;
5. import java.security.InvalidParameterException;
7. import org.winplus.serial.utils.SerialPort;
8. import org.winplus.serial.utils.SerialPortFinder;
10. import android.content.SharedPreferences;
12. public class MyApplication extends android.app.Application {
13. public SerialPortFinder mSerialPortFinder = new SerialPortFinder();
14. private SerialPort mSerialPort = null;
16. public SerialPort getSerialPort() throws SecurityException, IOException, InvalidParameterException {
17. if (mSerialPort == null) {
18. */\* Read serial port parameters \*/*
19. SharedPreferences sp = getSharedPreferences("android\_serialport\_api.sample\_preferences", MODE\_PRIVATE);
20. String path = sp.getString("DEVICE", "");
21. int baudrate = Integer.decode(sp.getString("BAUDRATE", "-1"));
23. */\* Check parameters \*/*
24. if ( (path.length() == 0) || (baudrate == -1)) {
25. throw new InvalidParameterException();
26. }
28. */\* Open the serial port \*/*
29. mSerialPort = new SerialPort(new File(path), baudrate, 0);
30. }
31. return mSerialPort;
32. }
34. public void closeSerialPort() {
35. if (mSerialPort != null) {
36. mSerialPort.close();
37. mSerialPort = null;
38. }
39. }
40. }

5） 新建一个继承抽象的Activity类，主要用于读取串口的信息

1. package org.winplus.serial;
3. import java.io.IOException;
4. import java.io.InputStream;
5. import java.io.OutputStream;
6. import java.security.InvalidParameterException;
8. import org.winplus.serial.utils.SerialPort;
10. import android.app.Activity;
11. import android.app.AlertDialog;
12. import android.content.DialogInterface;
13. import android.content.DialogInterface.OnClickListener;
14. import android.os.Bundle;
16. public abstract class SerialPortActivity extends Activity {
17. protected MyApplication mApplication;
18. protected SerialPort mSerialPort;
19. protected OutputStream mOutputStream;
20. private InputStream mInputStream;
21. private ReadThread mReadThread;
23. private class ReadThread extends Thread {
25. @Override
26. public void run() {
27. super.run();
28. while (!isInterrupted()) {
29. int size;
30. try {
31. byte[] buffer = new byte[64];
32. if (mInputStream == null)
33. return;
35. */\*\**
36. *\* 这里的read要尤其注意，它会一直等待数据，等到天荒地老，海枯石烂。如果要判断是否接受完成，只有设置结束标识，或作其他特殊的处理。*
37. *\*/*
38. size = mInputStream.read(buffer);
39. if (size > 0) {
40. onDataReceived(buffer, size);
41. }
42. } catch (IOException e) {
43. e.printStackTrace();
44. return;
45. }
46. }
47. }
48. }
50. private void DisplayError(int resourceId) {
51. AlertDialog.Builder b = new AlertDialog.Builder(this);
52. b.setTitle("Error");
53. b.setMessage(resourceId);
54. b.setPositiveButton("OK", new OnClickListener() {
55. public void onClick(DialogInterface dialog, int which) {
56. SerialPortActivity.this.finish();
57. }
58. });
59. b.show();
60. }
62. @Override
63. protected void onCreate(Bundle savedInstanceState) {
64. super.onCreate(savedInstanceState);
65. mApplication = (MyApplication) getApplication();
66. try {
67. mSerialPort = mApplication.getSerialPort();
68. mOutputStream = mSerialPort.getOutputStream();
69. mInputStream = mSerialPort.getInputStream();
71. */\* Create a receiving thread \*/*
72. mReadThread = new ReadThread();
73. mReadThread.start();
74. } catch (SecurityException e) {
75. DisplayError(R.string.error\_security);
76. } catch (IOException e) {
77. DisplayError(R.string.error\_unknown);
78. } catch (InvalidParameterException e) {
79. DisplayError(R.string.error\_configuration);
80. }
81. }
83. protected abstract void onDataReceived(final byte[] buffer, final int size);
85. @Override
86. protected void onDestroy() {
87. if (mReadThread != null)
88. mReadThread.interrupt();
89. mApplication.closeSerialPort();
90. mSerialPort = null;
91. super.onDestroy();
92. }
93. }

6）编写string.xml 以及baudrates.xml文件

    在string.xml文件中添加：

1. <string name="error\_configuration">Please configure your serial port first.</string>
2. <string name="error\_security">You do not have read/write permission to the serial port.</string>
3. <string name="error\_unknown">The serial port can not be opened for an unknown reason.</string>

在baudrates.xml文件中添加

1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
4. <string-array name="baudrates\_name">
5. <item>50</item>
6. <item>75</item>
7. <item>110</item>
8. <item>134</item>
9. <item>150</item>
10. <item>200</item>
11. <item>300</item>
12. <item>600</item>
13. <item>1200</item>
14. <item>1800</item>
15. <item>2400</item>
16. <item>4800</item>
17. <item>9600</item>
18. <item>19200</item>
19. <item>38400</item>
20. <item>57600</item>
21. <item>115200</item>
22. <item>230400</item>
23. <item>460800</item>
24. <item>500000</item>
25. <item>576000</item>
26. <item>921600</item>
27. <item>1000000</item>
28. <item>1152000</item>
29. <item>1500000</item>
30. <item>2000000</item>
31. <item>2500000</item>
32. <item>3000000</item>
33. <item>3500000</item>
34. <item>4000000</item>
35. </string-array>
36. <string-array name="baudrates\_value">
37. <item>50</item>
38. <item>75</item>
39. <item>110</item>
40. <item>134</item>
41. <item>150</item>
42. <item>200</item>
43. <item>300</item>
44. <item>600</item>
45. <item>1200</item>
46. <item>1800</item>
47. <item>2400</item>
48. <item>4800</item>
49. <item>9600</item>
50. <item>19200</item>
51. <item>38400</item>
52. <item>57600</item>
53. <item>115200</item>
54. <item>230400</item>
55. <item>460800</item>
56. <item>500000</item>
57. <item>576000</item>
58. <item>921600</item>
59. <item>1000000</item>
60. <item>1152000</item>
61. <item>1500000</item>
62. <item>2000000</item>
63. <item>2500000</item>
64. <item>3000000</item>
65. <item>3500000</item>
66. <item>4000000</item>
67. </string-array>
69. </resources>

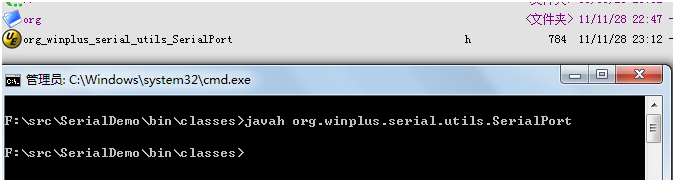
7）开始编写界面了：在main.xml布局文件中添加两个编辑框，一个用来发送命令，一个用来接收命令：

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="fill\_parent"
4. android:layout\_height="fill\_parent"
5. android:orientation="vertical" >
7. <EditText
8. android:id="@+id/EditTextReception"
9. android:layout\_width="fill\_parent"
10. android:layout\_height="fill\_parent"
11. android:layout\_weight="1"
12. android:gravity="top"
13. android:hint="Reception"
14. android:isScrollContainer="true"
15. android:scrollbarStyle="insideOverlay" >
16. </EditText>
18. <EditText
19. android:id="@+id/EditTextEmission"
20. android:layout\_width="fill\_parent"
21. android:layout\_height="wrap\_content"
22. android:hint="Emission"
23. android:lines="1" >
24. </EditText>
26. </LinearLayout>

8） SerialDemoActivity类的实现：

1. package org.winplus.serial;
3. import java.io.IOException;
5. import android.os.Bundle;
6. import android.view.KeyEvent;
7. import android.widget.EditText;
8. import android.widget.TextView;
9. import android.widget.TextView.OnEditorActionListener;
11. public class SerialDemoActivity extends SerialPortActivity{
12. EditText mReception;
14. @Override
15. protected void onCreate(Bundle savedInstanceState) {
16. super.onCreate(savedInstanceState);
17. setContentView(R.layout.main);
19. *// setTitle("Loopback test");*
20. mReception = (EditText) findViewById(R.id.EditTextReception);
22. EditText Emission = (EditText) findViewById(R.id.EditTextEmission);
23. Emission.setOnEditorActionListener(new OnEditorActionListener() {
24. public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
25. int i;
26. CharSequence t = v.getText();
27. char[] text = new char[t.length()];
28. for (i=0; i<t.length(); i++) {
29. text[i] = t.charAt(i);
30. }
31. try {
32. mOutputStream.write(new String(text).getBytes());
33. mOutputStream.write('\n');
34. } catch (IOException e) {
35. e.printStackTrace();
36. }
37. return false;
38. }
39. });
40. }
42. @Override
43. protected void onDataReceived(final byte[] buffer, final int size) {
44. runOnUiThread(new Runnable() {
45. public void run() {
46. if (mReception != null) {
47. mReception.append(new String(buffer, 0, size));
48. }
49. }
50. });
51. }
52. }

    写到这里，代码基本上写完了。下面就是要实现JNI层的功能了，要实现JNI，必须首先生成头文件，头文件的生成方式也很简单， 我们编译工程，在终端输入 javah org.winplus.serial.utils.SerialPort 则会生成头文件：org\_winplus\_serial\_utils\_SerialPort.h,这个头文件的名字可以随意命名。我们将它命名为：SerialPort.h拷贝到新建的目录jni中，新建SerialPort.c 文件，这两个文件的代码就不贴出来了。直接到上传的代码中看吧。



（四）串口的应用，可实现扫描头，指纹识别等外围USB转串口的特色应用。