**【android入门练习】**

**(注意AndroidManifest.xml中设置启动的Activity)**

1. **展示“Hello world”**

**（com.freedom.helloworld.FirstActivity）**

**目的：了解eclipse开发Android app环境；android项目文件结构；**

**要点：**

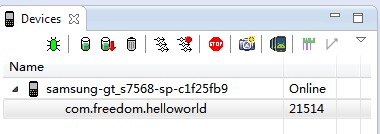
**. 模拟器的建立，能在上跑**

**. 能在真机上跑**

**输出：在手机启动app，显示"Hello world"**

****

备注1：DDMS->devices列表中可以看见连接上的真机型号等信息（注意手机通过USB连接，驱动安装不成功就没法在此列表中显示，说明连接不到设备）

****

备注2：也可以通过命令查看设备是否连接成功（前提：配置SDK环境变量path）

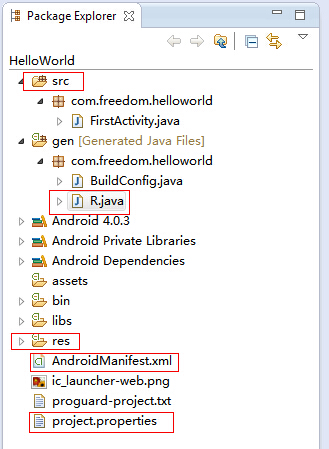


Adb devices :获取设备标识

**并且能回答以下问题：**

1. **project的文件夹src，gen，res，以及配置文件AndroidManifest.xml，project.properties作用**

**目录结构图：**



* src：源码目录（Activity等Java类）。
* gen：自动生成的目录，包含一些自动生成文件，如R.java（不需要手工编写）
* assets：该目录文件不会被编译成二进制编码，多用于放音视频，文本等原始格式文件。
* bin：用于存放编译后的Java文件及apk文件。
* libs：存放Android的源码包或需要引入的源码包。
* res：程序资源的所在目录，包括图片，数值等各类资源。
* AndroidManifest.xml：程序配置文件，它向系统描述了程序的信息。
* Project.properties:自动生成文件，主要是声明SDK的版本

1. **入口“main”在哪里**

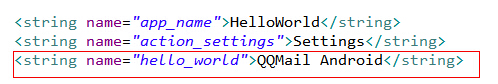
在AndroidManifest.xml文件中声明Activity带有<intent-filter>的Activity作为APP的入口。



**3. UI界面与逻辑代码，如何分割；如果要显示"QQMail Android"，怎么在原有代码修改？**

UI界面主要在layout的XML中定义，逻辑代码主要放在SRC下

修改：直接修改res/values目录下的String.xml即可



****

**附录：一些开发常用操作：**

**. 代码使用Utf8编码**

**Project -> Properties --> Text file encoding修改为UTF-8、**

**. 快捷键增修改**

**Window --> Properties --> General --> Keys**

**. 排版代码**

**Source --> Format (Ctrl+Shift+H)**

**. 自动整理import**

**Source --> Organize Imports (Ctrl+Shift+O)**

**. 代码跳转**

**Ctrl+点击**

**Alt+Left**

**Alt+Right**

**. 查找字符串：**

**Search --> Search... (Ctrl+H)**

**. 找到函数或变量被使用的地方：**

**Search --> References --> Project (或者References --> Project)**

**1.1 对1的结果，试用常用命令**

**目的：使用adb做一些常用操作；了解app在手机的目录结构**

**要点：**

**. 执行adb devices，说说结果是什么意思**

adb devices：获取设备标识

**. 执行adb kill-server，用于连接机器出现问题**

adb kill-server：关闭adb服务

**. 执行adb uninstall com.example.helloworld（上面app的名称）**

adb uninstall com.example.helloworld:卸载模拟器或者真机上的HelloWorld

**. 执行adb install HelloWorld.apk（上面app的名称）**

adb install HelloWorld.apk：安装HelloWorld.apk到模拟器或者手机上

**. 执行adb shell**

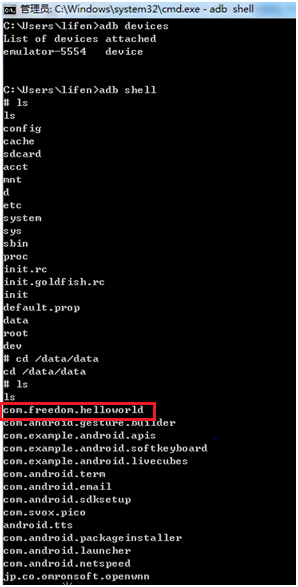
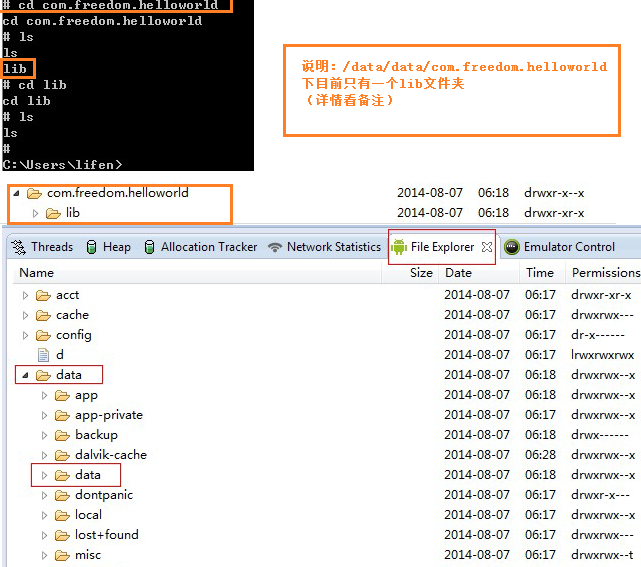
进入设备命令行模式（Linux下的命令都适用）

**. cd /data/data; ls，说说结果是什么意思**

cd /data/data：打开data文件夹下的data文件夹

ls：列出当前文件夹下的所有文件和文件夹

**. 如何进入上面1的app，看看里面目录有哪些，存放什么数据（需要手机root）**

** **

备注:/data/data/package\_name/ 应用的数据目录，包括cache、databases、lib、shared\_prefs，分别存放cache、数据库、lib、SharedPreferences数据  
/data/system/dropbox 存放系统fc，应用fc，应用ANR，系统启动日志、日志备份等。

**. 常用的linux命令：pwd（获取当前目录）, ls，chmod（修改读写权限）**

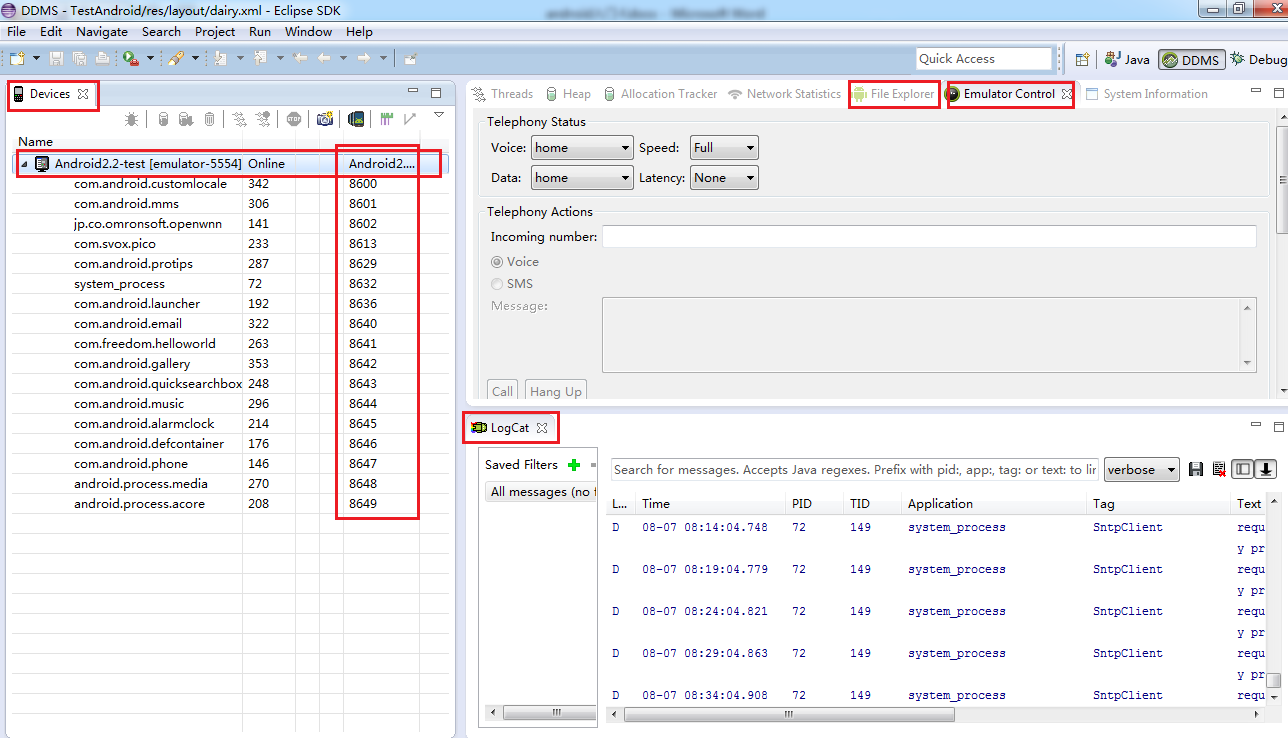
**. adb push，把pc的一个文件拷贝到手机里**

**. adb pull， 把手机的文件拷贝到pc**

**. 试用DDMS工具，ADB的可视化工具**

**DDMS工具:**（ Dalvik Debug Monitor Service）Android 开发环境中的Dalvik[虚拟机](http://baike.baidu.com/view/1132.htm)调试监控服务。

功能：为测试设备截屏，针对特定的进程查看正在运行的线程以及堆信息、Logcat、广播状态信息、模拟电话呼叫、接收SMS、虚拟地理坐标等等。

****

**工作原理：**

DDMS将搭建起IDE与测试终端(Emulator或者connected device)的链接,他们应用各自独立的端口监听调试信息,DDMS可以实时监测到测试终端的连接情况.当有新的测试终端连接后,DDMS将捕捉到终端的ID,并通过adb建立调试器,从而实现发送指令到测试终端的目的;DDMS监听第一个终端APP进程的端口为8600,App进程将分配8601,如果有更多的终端或者更多App进程将按照这个顺序依次类推.DDMS通过8700端口接收所有终端的指令.

**部分介绍：**

**Devices:**在这个面板可以看到所有与DDMS连接的终端的信息,以及每个终端正在运行的App进程,每个进程的右边相对应的是与调试器链接的端口,因为Android是基于Linux内核开发的操作平台,同时也保留了Linux中特有的进程ID,它介于进程名和端口号之间;

**Emulator Control:**通过这个面板的一些功能可以非常容易的使测试终端模拟真实手机所具备的一些交互功能比如:接听电话,根据选项模拟各种不同网络情况,模拟接受SMS消息和发送虚拟地址坐标用于测试GPS功能等;

**FileExplore:**刚才在上面有演示，就是查看真机或者模拟器上的文件夹或文件

**LogCat：**显示输出的调试信息

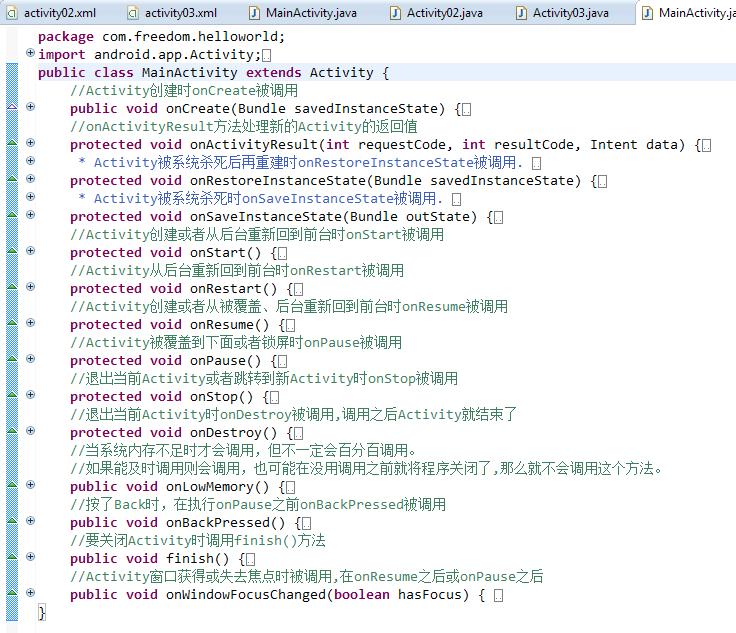
**ADB的可视化工具:**（Android Debug Bridge）Android调试桥

备注：暂时还没用过，基本cmd上都能满足需求

**1.2 在1基础上，补全Activity的生命周期接口onCreate, onRestart, onStart, onResume, onPause, onStop, onDestroy，并打logcat日志**

**目的：Logcat日志的使用；Activity生命周期**

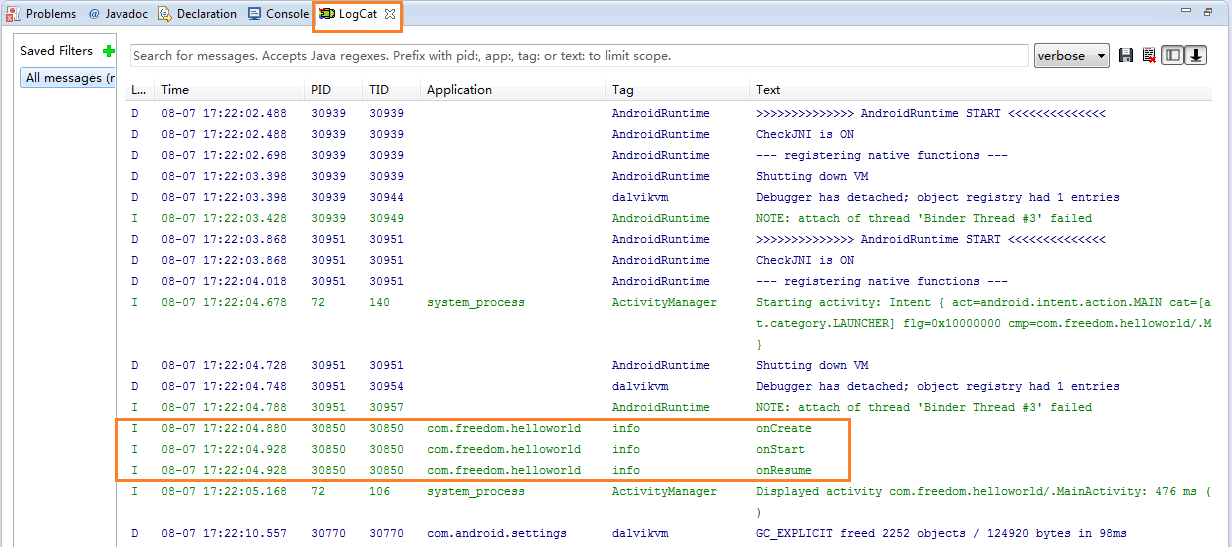
**（com.freedom.helloworld.MainActivity）**

****

**实验后，需要回答：**

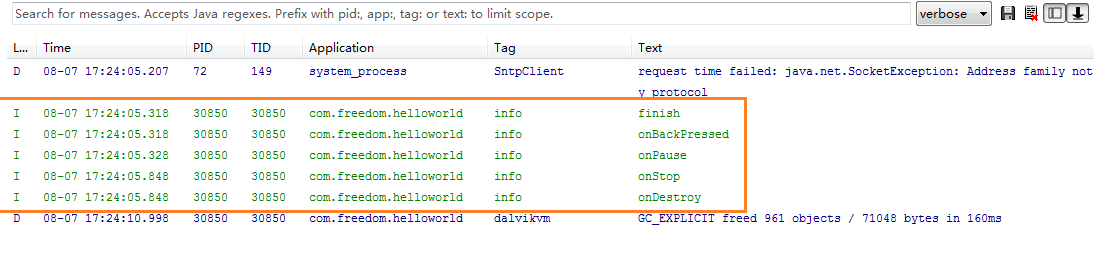
**1. 从icon第一次启动app，生命周期接口调用顺序**

**onCreate 🡪 onStart 🡪 onResume**

****

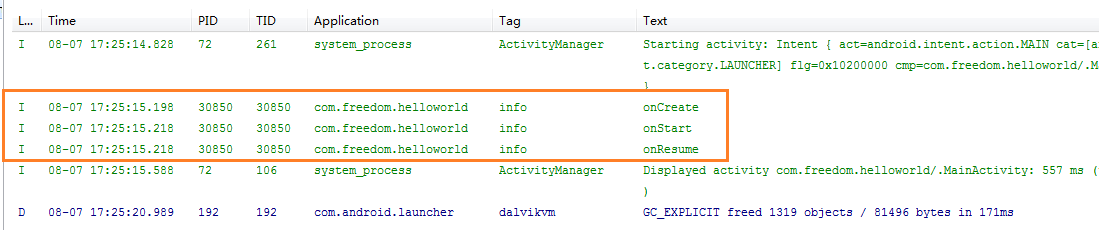
**2. 按返回，又如何**

**Finish 🡪 onBackPressed 🡪 onPause 🡪 onStop 🡪 onDestory**

****

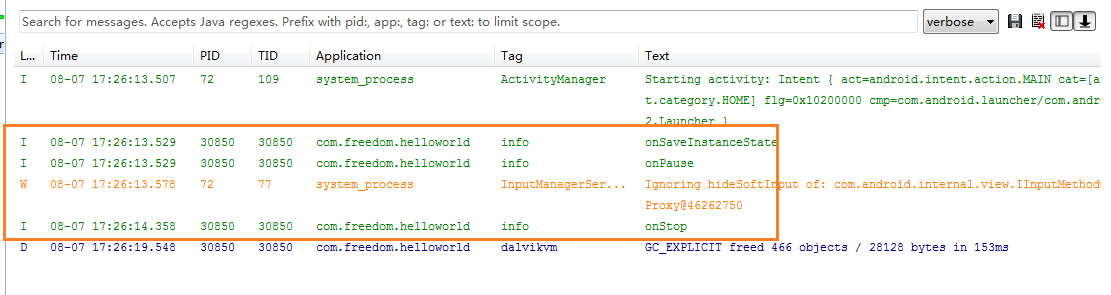
**3. 再按icon，又如何**

**onRestart 🡪 onStart 🡪 onResume**

****

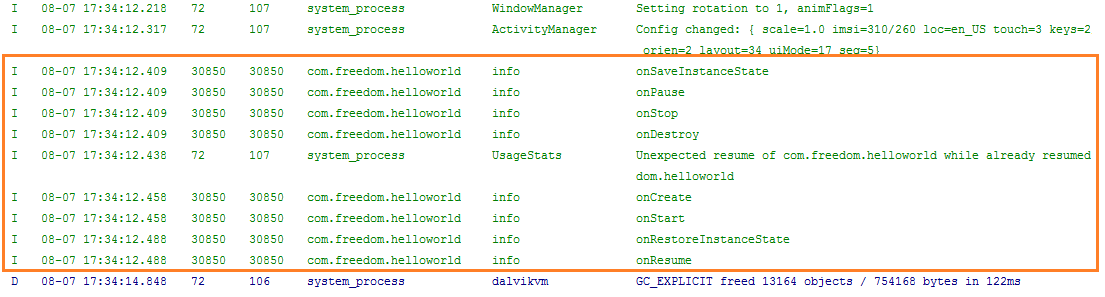
**补充：**

1. **按home键以后  
   onSaveInstanceState 🡪 onPause 🡪 onStop**

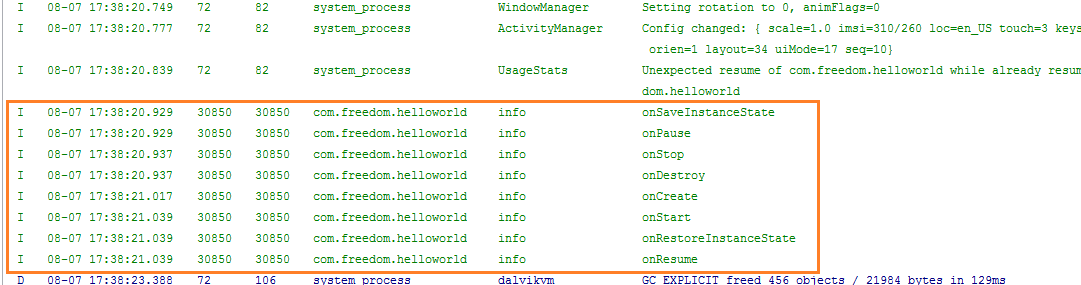


**5. 由竖屏转横屏（Ctrl+F12）**

**onSaveInstanceState 🡪 onPause 🡪 onStop 🡪 onDestory  
🡪 onCreate 🡪 onStart 🡪 onRestoreInstanceState 🡪 onResume**

****

**6. 横屏变竖屏  
onSaveInstanceState 🡪 onPause 🡪 onStop 🡪 onDestory 🡪 onCreate  
🡪 onStart 🡪 onRestoreInstanceState 🡪 onResume**



**备注：如果什么都不干只是横竖屏切换，相当于Activity被杀死又起来**

**7.** **android中还有一个方法onLowMemory**

**//当系统内存不足时才会调用，但不一定会百分百调用。如果能及时调用则会调用，也可能在没用调用之前就将程序关闭了,那么就不会调用这个方法。**

**1.3 在1.2基础上，新增两个activity，并在这三个activity上面增加3个按钮，跳到其他activity；log上能显示；还有一个按钮是返回，从哪个activity切换来的**

**目的：Intent使用；匿名类使用；事件的了解；Activity.finish；Activity的堆栈；Activity生命周期；**

**要点：返回键（back)，主页键（home），究竟返回到那个activity**

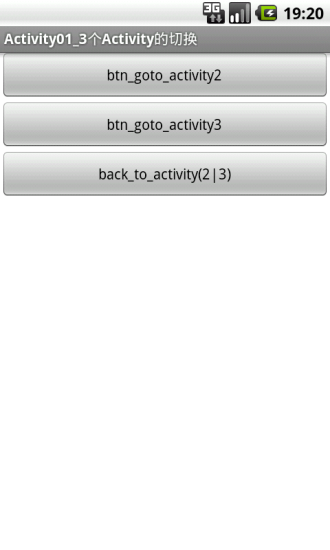
**输出：显示有三个页面**

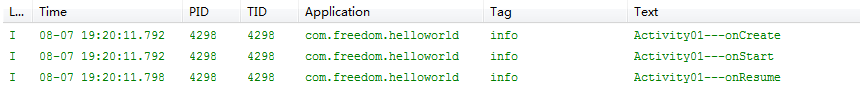
**Activity1**

**btn\_goto\_activity2**

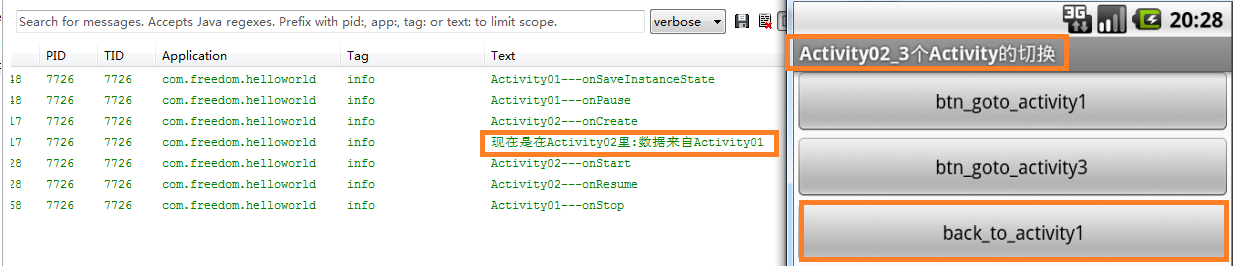
**btn\_goto\_activity3**

**back\_to\_activity(2|3) 具体是从哪里调进来**

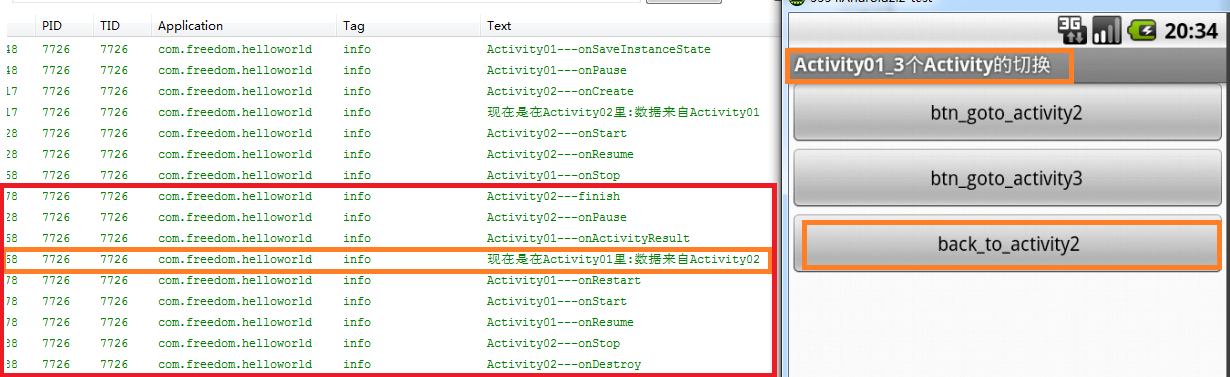
****

****

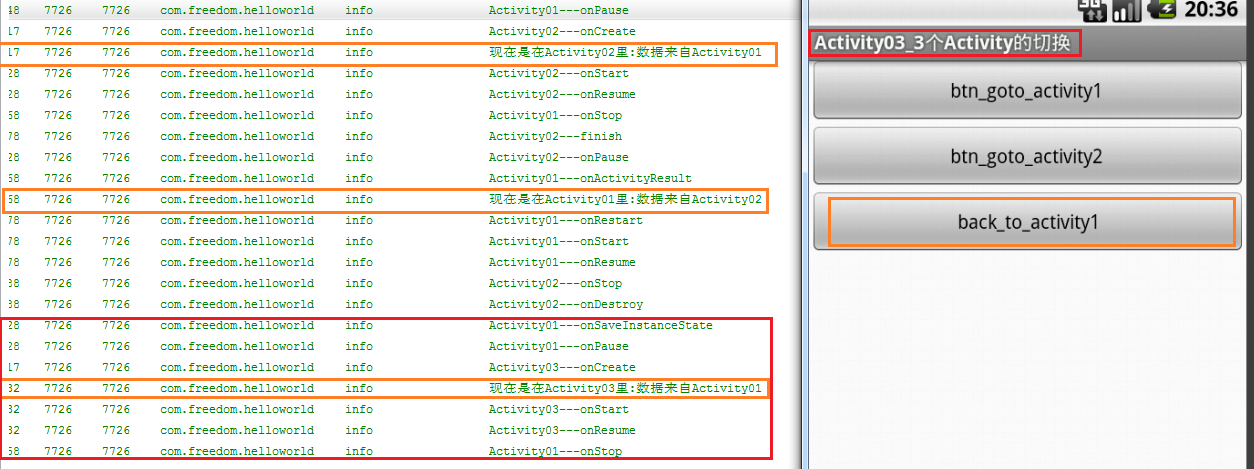
**点击“btn\_goto\_activity2”**

****

**点击“back\_to\_activity1”**

****

**点击“btn\_goto\_activity3”**

****

**点击“back\_to\_activity1”**

****

**……**

**Activity2**

**btn\_goto\_activity1**

**btn\_goto\_activity3**

**back\_to\_activity(1|3) 具体是从哪里调进来**

**道理同Activity1**

**Activity3**

**btn\_goto\_activity1**

**btn\_goto\_activity2**

**back\_to\_activity(1|2) 具体是从哪里调进来**

**道理同Activity1**

**实验后，需要回答：生命周期接口调用顺序（详细看Logcat截图）**

**1.4 在1.3基础上，选择其中一个activity，增加ListView，显示：**

**Activity1**

**btn\_goto\_activity2**

**btn\_goto\_activity3**

**back\_to\_activity(2|3) 具体是从哪里调进来**

**Label1**

**Item1.1**

**Item1.2**

**Label2**

**Item2.1**

**Item2.2**

**...**

**Label20**

**Item20.1**

**Item20.2**

**并且Label不接受点击；Item\*.1点击跳到一个activity；Item\*.2点击跳到另一个activity；**

**要求使用BaseAdapter**

**目的：ListView使用；Adapter如何显示数据（称为显存）；Adapter原理**

**要求：**

**. 数据类：**

**Label {**

**String name;**

**}**

**Item {**

**String labelName;**

**Strnig name;**

**}**

**细节：做的时候逐渐理解**

**输出：上面显示**

**1.5 在1.4，数据是从DB读出**

**目的：sqlite的使用；表的设计；insert数据；select数据；**

**要点：**

**. CreateTable**

**. 第一次启动，初始化，droptable，insert数据**

**. 获取数据，select**

**. 删除操作，delete**

**输出，显示跟1.4结果**

**1.6 在1.5上，优化，如果Label从20扩展到5000，增加初始化数据btn**

**目的：Cursor直出Adapter；繁重操作放到后台；后台与UI如何交互**

**显示**

**Activity1**

**btn\_goto\_activity2**

**btn\_goto\_activity3**

**back\_to\_activity(2|3) 具体是从哪里调进来**

**btn\_reset**

**Label1**

**Item1.1**

**Item1.2**

**Item1.3**

**Label2**

**Item2.1**

**Item2.2**

**Item2.3**

**...**

**Label5000**

**Item5000.1**

**Item5000.2**

**Item5000.3**

**问题：如果正在执行select，点击back\_to\_activity，有什么效果**