资料由 ltzx012 整理，感谢 superjunjin 的无私的奉献：
http://blog.csdn.net/superjunjin/article/category/1192401

第一章 Android 最全面试题 71 道题 详解

1. 下列哪些语句关于内存回收的说明是正确的? (b)

A、 程序员必须创建一个线程来释放内存

B、 内存回收程序负责释放无用内存

C、 内存回收程序允许程序员直接释放内存

D、 内存回收程序可以在指定的时间释放内存对象

android 内存回收机制

我想每个人第一次用 Android 的时候,不可避免的会去装 个任务管理器,然后对里面时刻都停留着一大堆的程序表 示触目惊心,然后会在桌面上建立一个快捷清空内存的按 钮,时不时啪的按一下,看着内存剩余数量从30多变成100 多然后很有快感... 其实吧,Android 是

Linux 的内核,每一个程序都是一个独立 的 JAVA 虚拟机,就和油汤里的油花一样互不干扰,这样充分 保证了万一某个程序的 JAVA 虚拟机崩溃,系统依旧稳定正 常运行. 而 Android 和传统 Linux 不一样的地方又在于,传统 Linux 在 进程活动停止后就结束了,这就类似于我们用 S60和 WM 一 样,关闭程序,内存释放.而 Android 会把这些进程保留在内 存里,干嘛呢?为了保证你再次激活这些进程时候启动的更 快,比如说我们挂在桌面的 Widgets, 具体一点我们拿新浪微博举例吧.我刚看完,退出,突然我想 我发一条微博吧, 那么这个时候我可以直接在桌面 Widgets 上操作---设想一 下如果我退出的时候这个进程就终止了,那么我在桌面上 点击 Widgets 的时候会不会卡顿一下甚至没有响应? ---这就跟我们把 Widgets 挂在桌面的行为完全背离了,放 在桌面上就是为了能随时观察到程序运行的情况, 以及随时可以快速调用程序.所以 Android 并没有在进程活 动停止就释放对应的内存.那么也许你还是会有疑问,那么 内存够不够用呢?

512的内存被我用的只剩56M 是不是很恐怖?其实系统一 点也不卡的,蛋定蛋定 是的,我理解,因为大家这么多年 Windows 都用习惯 了,Windows 内存不足的时候机器卡的会让你想砸掉机箱, 而且调用虚拟内存的时候硬盘喀喀喀想的让你肉疼. 你肯定也会怕你的手机明明 512M 内存结果就剩下30来 M 把你卡到崩溃.事实上呢,Android 会在系统需要更多内存 的时候,去释放掉那些占用内存的进程---这个活动是智能的.最早大家认为是有个排序,比如最近使 用过哪些程序(LRU 机制,Last Recently Used),然 后结束最早的进程.不过并非如此,否则就变成我们上小学 时候那样,个子高的块头大的男生跟班长下去拔草扛新书, 女生们留在班里绣花吧... 这样很明显不公平而且没准会结束掉那些我们并不想结束 掉的进程----譬如说这会儿我想切回到刚才后台的网页继 续浏览结果悲怆的发现它被系统给我强制关闭了...

Android 把进程分成了一些优先级,比如 前台进程(Foreground),比如我们正在看书,那么看书 的程序就是前台进程,这些进程是不会被系统优先结束的. 当我把它切到后台的时候,它就变成后台进程了. 还有可见进程(Visible),这个怎么说呢,譬如输入法程 序,你平时是看不见它的,但是在你打开输入界面的时候,它 会很快的弹出来,而不是让你等啊等啊等,看不到的原因是 透明度的机制,咱就不要钻牛角尖讨论为啥我看不见了... 还有桌面的 Widgets,比如我们的桌面时钟,这个东西就是 可见的,如果它被系统终止了会有什么样的结果?这个 Widgets 依然会显示在桌面上,但是时针不走了... 主要服务,比如说,电话的拨号功能,你也不想正急着打 电话呢结果人家给你卡半天吧,尤其像我这样联系人上 2000的,载入一遍真的很慢啊...所以这些主要服务平时也 不会被系统自动结束,除非你非要关它,关了也会自己重新 加载的.这也是你完全释放内存以后过一会就看着内存可 用值又慢慢降低的原因.

次要服务(secondary server),诸如谷歌企业套 件,Gmail,联系人,看着这些程序出现在任务管理器里可能 你会非常的莫名其妙,丫的这都哪跟哪啊我没开啊...其实 它们和一些系统功能也是息息相关的,比如 Gmail 的邮件推 送,我们时常需要用到它们,所以系统也太会去终止它们.甚 至于 HTC 机器上著名的 HTC Sense,这个也是次要服务,但 是其实它承接着整个系统

界面的运行,所以,如果你强行关 闭所有进程的时候,你的屏幕会变成一片白...然后慢慢等 HTC Sense 加载.

后台进程(hidden),就是我们通常意义上理解的启动后 被切换到后台的进程,比如如浏览器和阅读器.后台进程的 管理策略有多种,但是一般来讲,系统都会视内存情况,尽可 能多的保留后台程序,这样会影响到你启动别的程序的运 行速度---我想这个很好理解,因为内存确实不够了,而且你 还没让系统自动释放内存.但好处是,你再次切换到这些已 启动的程序时几乎是无缝的,速度绝对比你从0开始启动它 要快得多.所以,这种后台进程在内存极度不够的时候,肯定 会被系统选择性的干掉的. 内容供应节点(content provider),没有程序实体,仅提供内容供别的程序去用的,比如日历供应节点,邮件供应节 点等.在系统自动终止进程时,这类程序享有优先的被干掉 权... 空进程(empty), 没有任何东西在内运行的进程,有些程序在退出后,依然会 在进程中驻留一个空进程,这个进程里没有任何数据在运 行,作用往往是提高该程序下次的启动速度或者记录程序 的一些历史信息.这部分进程无疑是系统最先终止的.

说了这么多,其实还是要结合实际的程序来看一下的,比如 Android 这个很有名的自动内存调配的软件,Auto Memory Manager,它的设置和帮助界面就如上面所说的, 它自动提供了多种默认配置,例如极速模式,这个模式下,会 帮助你在设定好的临界值区间上,结束空进程以及内容供 应节点等等低优先级保留权的进程,来给你腾出更多的内 存,加速新运行程序打开的速度,但是它也说明了这种模式 的弊端,就是一些可能你不想被关闭的进程会被过早的关 闭,比如说,闹钟---在 G2 G3还很火爆的2009年,很多用户 在买完手机后给我抱怨,哎呀这个机器闹钟怎么老不响 啊...上班老迟到...其实这就是因为手动结束进程的时候结 果把闹钟也给干掉了.系统的时间是会一直走的,这属于主 要服务,而闹钟呢,只是主要服务的一个附属品,所以被结束 后,是不会自动被启动的,既然没有启动自然就不会响了.与 此类似的例子就是里程碑不充电的 BUG,这是因为 Moto 的 机器里有个 USB 的进程,如果你把它结束后,理论上会重新 启动的但是也会不启动,后面这种情况出现的结果就是你 插充电器没反应,插数据线连电脑没反应...重启手机就好 了. 当然我知道大家的洁癖很多,有的人就是见不得内存值太 小...好吧如果你不想一些被系统认为不太重要而你又很需 要的进程被你自己亲手扼杀的话,那么我推荐你使用高级 任务管理器这个程序,你可以把一些进程自动隐藏起来,也 就是说当你挥起狼牙棒横扫一堆进程的时候,你设置好的 几个进程是不会受任何影响的,比如桌面 Launcher,比如闹 钟,比如 USB,等等等等.但话说回来,我是不建议大家去手动 管理 Android 的内存,也许你会不习惯---我也没啥好劝告 的,总之,不要把你的智能机想的那么笨就行了. 刚才全杀掉进程后,过了一会,我的 DEFY 又变成剩余60M 内 存,还是没啥鸭梨啊...如果你感兴趣可以做个试验,内 存很少的时候,你打开一个大游戏,然后退出,你会发现...

http://reedhistudy.diandian.com/post/2011-09-15/5045645

2. 下面异常是属于 Runtime Exception 的是（abcd）(多选)

A、ArithmeticException

B、IllegalArgumentException

C、NullPointerException

D、BufferUnderflowException

A、ArithmeticException

当出现异常的运算条件时，抛出此异常。例如，一个整数“除以零”时，抛出此类的一个实例。

B、IllegalArgumentException

抛出的异常表明向方法传递了一个不合法或不正确的参数。

C、NullPointerException

D、BufferUnderflowException （不明白，没碰到过）

编码问题导致 java_BufferUnderflowException 异常

公共类 BufferUnderflowException 的

延伸的 RuntimeException

未经检查的异常时，抛出一个相对 get 操作达到源缓冲区的限制。

3. Math.round(11.5)等于多少(). Math.round(-11.5)等于多少(c).

A、11 ,-11　　B、11 ,-12　　C、12 ,-11　　D、12 ,-12

四舍五入 四和五是指正的4，5

-11.5 这么看 -11.5 = -12 +0.5 ，0.5按四舍五入为1 ，-12+1 = -11，所以

Math.round(-11.5)==-11

-0.5 = -1 + 0.5　　0.5按四舍五入为1 ，-1+1 = 0，所以 Math.round(-0.5)==0

11.5 四舍五入 显然 Math.round(11.5)==12

round 方法返回与参数最接近的长整数，参数加0.5后求其 floor（小于等于该数的最大整数）

4. 下列程序段的输出结果是：（b）

```
void complicatedexpression_r(){
int x=20, y=30;
boolean b;

b=x>50&&y>60||x>50&&y<-60||x<-50&&y>
60||x<-50&&y<-60;
System.out.println(b);
}
```

A、true　B、false　C、1　D、011.activity

&&（与）的优先级比||（或）高

5. 对一些资源以及状态的操作保存，最好是保存在生命周期的哪个函数中进行(d)

A、onPause()　B、onCreate()　C、 onResume()　D、onStart()

Activity 详解（生命周期、以各种方式启动 Activity、状态保存，完全退出等）

http://blog.csdn.net/tangcheng_ok/article/details/6755194

6. Intent 传递数据时，下列的数据类型哪些可以被传递（abcd）(多选)

A、Serializable　B、charsequence　C、Parcelable　D、Bundle

android 数据传递详解（Serialization、Parcelable、Parcel、Intent、Bundle）

http://jojol-zhou.iteye.com/blog/1401905

Android 中 Intent 传递对象的两种方法(Serializable,Parcelable)

http://blog.csdn.net/xyz_lmn/article/details/5908355

7. android 中下列属于 Intent 的作用的是(c)

A、实现应用程序间的数据共享

B、是一段长的生命周期，没有用户界面的程序，可以保持应用在后台运行，而不会因为切换页面而消失

C、可以实现界面间的切换，可以包含动作和动作数据，连接四大组件的纽带

D、处理一个应用程序整体性的工作

8. 下列属于 SAX 解析 xml 文件的优点的是(b)

A、将整个文档树在内存中，便于操作，支持删除，修改，重新排列等多种功能（dom 解析优点）

B、不用事先调入整个文档，占用资源少（sax 解析优点）

C、整个文档调入内存，浪费时间和空间（dom 解析缺点）

D、不是长久驻留在内存，数据不是持久的，事件过后，若没有保存数据，数据就会（sax 解析缺点）

消失

不需要像 dom 解析那样在内存中建立一个 dom 对象，占用内存，sax 解析是逐行解析的，每次读入内存的只是一行 xml，所以速度快，效率高点。不过 sax 一般是处理固定格式的 xml。

9. 下面的对自定 style 的方式正确的是（a）

A、 <resources>

<style name="myStyle">
<item name="android:layout_width">fill_parent</item>
</style>
</resources>

B、 <style name="myStyle">

<item name="android:layout_width">fill_parent</item>（没有<resources>）
</style>

C、 <resources>

<item name="android:layout_width">fill_parent</item>（没有</style>）
</resources>

D、 <resources>

<style name="android:layout_width">fill_parent</style>（</style>应为</item>）
</resources>

10. 在 android 中使用 Menu 时可能需要重写的方法有（ac）。(多选)

A、onCreateOptionsMenu()

B、onCreateMenu()

C、onOptionsItemSelected()

D、onItemSelected()

```
 //当客户点击 MENU 按钮的时候，调用该方法
      @Override
public boolean onCreateOptionsMenu(Menu menu) {
      menu.add(0, 1, 1, R.string.exit);
      menu.add(0,2,2,R.string.about);
return super.onCreateOptionsMenu(menu);
}
      //当客户点击菜单当中的某一个选项时，会调用该方法
@Override
public boolean onOptionsItemSelected(MenuItem item) {
if(item.getItemId() == 1){
finish();
}
return super.onOptionsItemSelected(item);
}
```

11. 在 SQL Server Management Studio 中运行下列 T-SQL 语句，其输出值（c）。

```
SELECT @@IDENTITY
```

A、可能为0.1

B、可能为3

C、 不可能为-100

D、肯定为0

@@identity 是表示的是最近一次向具有 identity 属性(即自增列)的表插入数据时对应的自增列的值，是系统定义的全局变量。一般系统定义的全局变量都是以@@开头，用户自定义变量以@开头。比如有个表 A，它的自增列是 id，当向 A 表插入一行数据后，如果插入数据后自增列的值自动增加至101，则通过 select @@identity 得到的值就是101。使用@@identity

的前提是在进行 insert 操作后，执行 select @@identity 的时候连接没有关闭，否则得到的将是 NULL 值。

12. 在 SQL Server 2005中运行如下 T-SQL 语句，假定 SALES 表中有多行数据，执行查询之 后的结果是（d）。

```
BEGIN TRANSACTION A
  Update SALES Set qty=30 WHERE qty<30
BEGIN TRANSACTION B
Update SALES Set qty=40 WHERE qty<40
Update SALES Set qty=50 WHERE qty<50
Update SALES Set qty=60 WHERE qty<60
COMMITTRANSACTION B
COMMIT TRANSACTION A
```

A、SALES 表中 qty 列最小值大于等于30

B、SALES 表中 qty 列最小值大于等于40

C、SALES 表中 qty 列的数据全部为50

D、SALES 表中 qty 列最小值大于等于60

Update SALES Set qty=60 WHERE qty<60（关键在最后一句，执行完数据就都是大于等于60了）

13. 在 android 中使用 SQLiteOpenHelper 这

个辅助类时，可以生成一个数据库，并可以对数据库版本进行管理的方法可以是(ab)

A、getWriteableDatabase()

B、getReadableDatabase()

C、getDatabase()

D、getAbleDatabase()

14. android 关于 service 生命周期的 onCreate()和 onStart()说法正确的是(ad)(多选题)

A、当第一次启动的时候先后调用 onCreate()和 onStart()方法

B、当第一次启动的时候只会调用 onCreate()方法

C、如果 service 已经启动，将先后调用 onCreate()和 onStart()方法

D、如果 service 已经启动，只会执行 onStart()方法，不在执行 onCreate()方法

15. 下面是属于 GLSurFaceView 特性的是(abc)(多选)

A、管理一个 surface，这个 surface 就是一块特殊的内存，能直接排版到 android 的视图 view 上。

B、管理一个 EGL display，它能让 opengl 把内容渲染到上述的 surface 上。

C、让渲染器在独立的线程里运作，和 UI 线程分离。

D、可以直接从内存或者 DMA 等硬件接口取得图像数据

android.opengl.GLSurfaceView 概述

http://blog.csdn.net/xqhrs232/article/details/6195824

GLSurfaceView 是一个视图，继承至 SurfaceView，它内嵌的 surface 专门负责 OpenGL 渲染。

GLSurfaceView 提供了下列特性：

1> 管理一个 surface，这个 surface 就是一块特殊的内存，能直接排版到 android 的视图 view 上。

2> 管理一个 EGL display，它能让 opengl 把内容渲染到上述的 surface 上。

3> 用户自定义渲染器(render)。

4> 让渲染器在独立的线程里运作，和 UI 线程分离。

5> 支持按需渲染(on-demand)和连续渲染(continuous)。

6> 一些可选工具，如调试。

16. 下面在 AndroidManifest.xml 文件中注册 BroadcastReceiver 方式正确的(a)

A、<receiver android:name="NewBroad">

<intent-filter>

<action

android:name="android.provider.action.NewBroad"/>

<action>

</intent-filter>

</receiver>

B、<receiver android:name="NewBroad">

<intent-filter>

android:name="android.provider.action.NewBroad"/>

</intent-filter>

</receiver>

C、<receiver android:name="NewBroad">

<action

android:name="android.provider.action.NewBroad"/>

<action>

</receiver>

D、<intent-filter>

<receiver android:name="NewBroad">

<action>

android:name="android.provider.action.NewBroad"/>

<action>

</receiver>

</intent-filter>

17. 关于 ContenValues 类说法正确的是(a)

A、他和 Hashtable 比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

名是 String 类型，而值都是基本类型

B、他和 Hashtable 比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

名是任意类型，而值都是基本类型

C、他和 Hashtable 比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

名，可以为空，而值都是 String 类型

D、他和 Hashtable 比较类似，也是负责存储一些名值对，但是他存储的名值对当

中

的名是 String 类型，而值也是 String 类型

18. 我们都知道 Hanlder 是线程与 Activity 通信的桥梁,如果线程处理不当，你的机器就会变得越慢，那么线程销毁的方法是(a)

A、onDestroy()

B、onClear()

C、onFinish()

D、onStop()

19. 下面退出 Activity 错误的方法是（c）

A、finish()

B、抛异常强制退出

C、System.exit()　System.exit(0) 0是正常退出

其他数字是表示不正常退出

D、onStop()

20. 下面属于 android 的动画分类的有(ab)(多项)

A、Tween　B、Frame C、Draw D、Animation

Android 动画模式

Animation 主要有两种动画模式：

一种是 tweened animation(渐变动画)

XML 中	JavaCode
alpha	AlphaAnimation
scale	ScaleAnimation

一种是 frame by frame(画面转换动画)

XML 中	JavaCode
translate	TranslateAnimation
rotate	RotateAnimation

21. 下面关于 Android dvm 的进程和 Linux

的进程,应用程序的进程说法正确的是(d)

A、DVM 指 dalivk 的虚拟机.每一个 Android 应用程序都在它自己的进程中运行,不一定拥有一个独立 的 Dalvik 虚拟机实例.而每一个 DVM 都是在 Linux 中的一个进程,所以说可以认为是同一个概念.

B、DVM 指 dalivk 的虚拟机.每一个 Android 应用程序都在它自己的进程中运行,不一定拥有一个独立的 Dalvik 虚拟机实例.而每一个 DVM 不一定都是在 Linux 中的一个进程,所以说不是一个概念.

C、DVM 指 dalivk 的虚拟机.每一个 Android 应用程序都在它自己的进程中运行,都拥有一个独立的 Dalvik 虚拟机实例.而每一个 DVM 不一定都是在 Linux 中的一个进程,所以说不是一个概念.

D、DVM 指 dalivk 的虚拟机.每一个 Android 应用程序都在它自己的进程中运行,都拥有一个独立的 Dalvik 虚拟机实例.而每一个 DVM 都是在 Linux 中的一个进程,所以说可以认为是同一个概念.

22. Android 项目工程下面的 assets 目录的作用是什么 b

A、放置应用到的图片资源。

B、主要放置多媒体等数据文件

C、放置字符串，颜色，数组等常量数据

D、放置一些与 UI 相应的布局文件，都是 xml 文件

23. 关于 res/raw 目录说法正确的是(a)

A、 这里的文件是原封不动的存储到设备上不会转换为二进制的格式

B、 这里的文件是原封不动的存储到设备上会转换为二进制的格式

C、 这里的文件最终以二进制的格式存储到指定的包中

D、 这里的文件最终不会以二进制的格式存储到指定的包中

24. 下列对 android NDK 的理解正确的是 (abcd)

A、 NDK 是一系列工具的集合

B、 NDK 提供了一份稳定、功能有限的 API 头文件声明。

C、 使“Java+C” 的开发方式终于转正，成为官方支持的开发方式

D、 NDK 将是 Android 平台支持 C 开发的开端

Windows 平台下如何使用 Android NDK

http://yuchen.blog.51cto.com/2739238/623472/

二．填空题

25. android 中常用的四个布局是

LinearLayout（线性布局）、FrameLayout（单帧布局）、RelativeLayout（相对布局）和 TableLayout（表格布局）

26. android 的四大组件是 activity，service，broadcast 和 Content Provider。

27. java.io 包中的 objectinputstream 和 objectoutputstream 类主要用于对对象(Object)的读写。

28. android 中 service 的实现方法是：startservice 和 bindservice。

Service 的生命周期方法比 Activity 少一些，只有 onCreate, onStart, onDestroy

我们有两种方式启动一个 Service,他们对 Service 生命周期的影响是不一样的。

1 通过 startService

Service 会经历 onCreate --> onStart

stopService 的时候直接 onDestroy

如果是 调用者 直接退出而没有调用 stopService 的话，Service 会一直在后台运行。

下次调用者再起来仍然可以 stopService。

2 通过 bindService

Service 只会运行 onCreate， 这个时候 调用者和 Service 绑定在一起

调用者退出了，Srevice 就会调用 onUnbind-->onDestroyed

所谓绑定在一起就共存亡了。

1. Started Service 中使用 StartService（）方法来进行方法的调用，调用者和服务之间没有联系，即使调用者退出了，服务依然在进行【onCreate()->onStartCommand()->startService()->onDestroy()】，注意其中没有 onStart()，主要是被 onStartCommand()方法给取代了，onStart 方法不推荐使用了。

2. BindService 中使用 bindService()方法来绑定服务，调用者和绑定者绑在一起，调用者一旦退出服务也就终止了【onCreate()->onBind()->onUnbind()->onDestroy()】。

29. activity 一般会重载7个方法用来维护其生命周期，除了 onCreate(),onStart(),onDestory() 外还有 onrestart,onresume,onpause,onstop。

30. android 的数据存储的方式 sharedpreference,文件,SQlite,contentprovider,网络。

1. 使用 SharedPreferences 存储数据；
2. 文件存储数据；
3. SQLite 数据库存储数据；
4. 使用 ContentProvider 存储数据；
5. 网络存储数据；

31. 当启动一个 Activity 并且新的 Activity 执行完后需要返回到启动它的 Activity 来执行的回调函数是 startActivityForResult

startActivityForResult(Intent,requestCode)//启动一个 activity 包含参数请求码和具体的 intent 数据，其中请求码可以用来识别子活动。

32. 请使用命令行的方式创建一个名字为

myAvd,sdk 版本为2.2,sd 卡是在 d 盘的根目录下,名字为 scard.img， 并指定屏幕大小 HVGA. _____android create acd -n myAvd -t 8 -s HVDA - C d:\card.img_____。

33. 程序运行的结果是：_____good and gbc_____。

```
public class Example{
String str=new String("good");
char[]ch={'a','b','c'};
public static void main(String args[]){
    Example ex=new Example();
    ex.change(ex.str,ex.ch);
    System.out.print(ex.str+" and ");
    Sytem.out.print(ex.ch);
}
public void change(String str,char ch[]){
    str="test ok";
    ch[0]='g';
}
}
```

34. 在 android 中，请简述 jni 的调用过程。(8分)

1)安装和下载 Cygwin，下载 Android NDK

2)在 ndk 项目中 JNI 接口的设计

3)使用 C/C++实现本地方法

4)JNI 生成动态链接库.so 文件

5)将动态链接库复制到 java 工程，在 java 工程中调用，运行 java 工程即可

35. 简述 Android 应用程序结构是哪些?（7分）

Android 应用程序结构是：

Linux Kernel(Linux 内核)、Libraries(系统运行库或者是 c/c++核心库)、Application Framework(开发框架包)、Applications(核心应用程序)

36. 请继承 SQLiteOpenHelper 实现：(10分)

1）.创建一个版本为1的“diaryOpenHelper.db”的数据库，

2）.同时创建一个 “diary” 表（包含一个_id 主键并自增长，topic 字符型100长度， content 字符型1000长度）

3）.在数据库版本变化时请删除 diary 表，并重新创建出 diary 表。

```
public class DBHelper    extends SQLiteOpenHelper {
public final static String DATABASENAME = "diaryOpenHelper.db";
public final static int DATABASEVERSION = 1;
//创建数据库
public DBHelper(Context context,String name,CursorFactory factory,int version)
{
super(context, name, factory, version);
}
//创建表等机构性文件
```

```
public void onCreate(SQLiteDatabase db)
{
String sql ="create table diary"+
"("+
"_id integer primary key autoincrement,"+
"topic varchar(100),"+
"content varchar(1000)"+
")";
db.execSQL(sql);
}
//若数据库版本有更新，则调用此方法

public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion)
{
String sql = "drop table if exists diary";
db.execSQL(sql);
this.onCreate(db);
}
}
```

37. 页面上现有 ProgressBar 控件 progressBar，请用书写线程以10秒的的时间完成其进度显示工作。（10分）

答案

```
public class ProgressBarStu extends Activity {
private ProgressBar progressBar = null;
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.progressbar);
//从这到下是关键

progressBar = (ProgressBar)findViewById(R.id.progressBar);
Thread thread = new Thread(new Runnable() {
@Override
```

```
public void run() {
int progressBarMax = progressBar.getMax();
try {
while(progressBarMax!=progressBar.getProgress())
{
int stepProgress = progressBarMax/10;
int currentprogress = progressBar.getProgress();
progressBar.setProgress(currentprogress+stepProgress);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
});
thread.start();
//关键结束

}
}
```

38. 请描述下 Activity 的生命周期。

必调用的三个方法：onCreate() --> onStart() --> onResume()，用 AAA 表示

（1）父 Activity 启动子 Activity，子 Actvity 退出，父 Activity 调用顺序如下

AAA --> onFreeze() --> onPause() --> onStop() --> onRestart() --> onStart(),onResume() …

（2）用户点击 Home，Actvity 调用顺序如下

AAA --> onFreeze() --> onPause() --> onStop() -- Maybe --> onDestroy() – Maybe

（3）调用 finish()， Activity 调用顺序如下

AAA --> onPause() --> onStop() --> onDestroy()

（4）在 Activity 上显示 dialog， Activity 调用顺序如下

AAA

（5）在父 Activity 上显示透明的或非全屏的 activity，Activity 调用顺序如下

AAA --> onFreeze() --> onPause()

（6）设备进入睡眠状态，Activity 调用顺序如下

AAA --> onFreeze() --> onPause()

39. 如果后台的 Activity 由于某原因被系统回收了，如何在被系统回收之前保存当前状态？

onSaveInstanceState()

当你的程序中某一个 Activity A 在运行时，主动或被动地运行另一个新的 Activity B，这个时候 A 会执行 onSaveInstanceState()。B 完成以后又会来找 A，这个时候就有两种情况：一是 A 被回收，二是 A 没有被回收，被回收的 A 就要重新调用 onCreate()方法，不同于直接启动的是这回 onCreate()里是带上了参数 savedInstanceState；而没被收回的就直接执行 onResume()，跳过 onCreate()了。

40. 如何将一个 Activity 设置成窗口的样式。

在 AndroidManifest.xml 中定义 Activity 的地方一句话

android:theme="@android:style/Theme.Dialog"或

android:theme="@android:style/Theme.Translucent"就变成半透明的

41. 如何退出 Activity？如何安全退出已调用

多个 Activity 的 Application？

对于单一 Activity 的应用来说，退出很简单，直接 finish()即可。

当然，也可以用 killProcess()和 System.exit()这样的方法。

但是，对于多 Activity 的应用来说，在打开多个 Activity 后，如果想在最后打开的 Activity 直接退出，上边的方法都是没有用的，因为上边的方法都是结束一个 Activity 而已。

当然，网上也有人说可以。

就好像有人问，在应用里如何捕获 Home 键，有人就会说用 keyCode 比较 KEYCODE_HOME 即可，而事实上如果不修改 framework，根本不可能做到这一点一样。

所以，最好还是自己亲自试一下。

那么，有没有办法直接退出整个应用呢？

在2.1之前，可以使用 ActivityManager 的 restartPackage 方法。

它可以直接结束整个应用。在使用时需要权限 android.permission.RESTART_PACKAGES。

注意不要被它的名字迷惑。

可是，在2.2，这个方法失效了。

在2.2添加了一个新的方法，killBackgroundProcesses()，需要权限 android.permission.KILL_BACKGROUND_PROCESSES。

可惜的是，它和2.2的 restartPackage 一样，根本起不到应有的效果。

另外还有一个方法，就是系统自带的应用程序管理里，强制结束程序的方法，forceStopPackage()。

它需要权限 android.permission.FORCE_STOP_PACKAGES。

并且需要添加 android:sharedUserId="android.uid.system"属性

同样可惜的是，该方法是非公开的，他只能运行在系统进程，第三方程序无法调用。

因为需要在 Android.mk 中添加 LOCAL_CERTIFICATE := platform。

而 Android.mk 是用于在 Android 源码下编译程序用的。

从以上可以看出，在2.2，没有办法直接结束一个应用，而只能用自己的办法间接办到。

现提供几个方法，供参考：

1、抛异常强制退出：

该方法通过抛异常，使程序 Force Close。

验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出 Force Close 的窗口。

2、记录打开的 Activity：

每打开一个 Activity，就记录下来。在需要退出时，关闭每一个 Activity 即可。

3、发送特定广播：

在需要结束应用时，发送一个特定的广播，每个 Activity 收到广播后，关闭即可。

4、递归退出

在打开新的 Activity 时使用 startActivityForResult，然后自己加标志，在 onActivityResult 中处理，递归关闭。

除了第一个，都是想办法把每一个 Activity 都结束掉，间接达到目的。

但是这样做同样不完美。

你会发现，如果自己的应用程序对每一个 Activity 都设置了 nosensor，在两个 Activity 结束的间隙，sensor 可能有效了。

但至少，我们的目的达到了，而且没有影响用户使用。

为了编程方便，最好定义一个 Activity 基类，处理这些共通问题。

42. 请介绍下 Android 中常用的五种布局。

FrameLayout（框架布局），LinearLayout （线性布局），AbsoluteLayout（绝对布局），

RelativeLayout（相对布局），TableLayout（表格布局）

43. 请介绍下 Android 的数据存储方式。

一.SharedPreferences 方式

二.文件存储方式

三.SQLite 数据库方式

四.内容提供器（Content provider）方式

五. 网络存储方式

44. 请介绍下 ContentProvider 是如何实现数据共享的。

创建一个属于你自己的 Content provider 或者将你的数据添加到一个已经存在的 Content

provider 中，前提是有相同数据类型并且有写入 Content provider 的权限。

45. 如何启用 Service，如何停用 Service。

Android 中的 service 类似于 windows 中的 service，service 一般没有用户操作界面，它运行

于系统中不容易被用户发觉，

可以使用它开发如监控之类的程序。

一。步骤

第一步：继承 Service 类

```
public class SMSService extends Service { }
```

第二步：在 AndroidManifest.xml 文件中的<application>节点里对服务进行配置:

```
<service android:name=".DemoService" />
```

二。Context.startService()和 Context.bindService

服务不能自己运行，需要通过调用 Context.startService()或 Context.bindService()方法启动服务。这两个方法都可

以启动 Service，但是它们的使用场合有所不同。

1.使用 startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。

使用 bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止。

2.采用 Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的

onCreate()方法，

接着调用 onStart()方法。如果调用 startService()方法前服务已经被创建，多次调用 startService()方法并

不会导致多次创建服务，但会导致多次调用 onStart()方法。

采用 startService()方法启动的服务，只能调用 Context.stopService()方法结束服务，服务结束时会调用

onDestroy()方法。

3.采用 Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的

onCreate()方法，

接着调用 onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的 onUnbind()方法，

。接着调用 onDestroy()方法。如果调用 bindService()方法前服务已经被绑定，多次调用 bindService()方法并不会

导致多次创建服务及绑定(也就是说 onCreate()和 onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务

解除绑定，可以调用 unbindService()方法，调用该方法也会导致系统调用服务的 onUnbind()-->onDestroy()方法。

三。Service 的生命周期

1.Service 常用生命周期回调方法如下：

onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次 startService()或 bindService()方法，

服务也只被创建一次。 onDestroy()该方法在服务被终止时调用。

2. Context.startService()启动 Service 有关的生命周期方法

onStart() 只有采用 Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。

多次调用 startService()方法尽管不会多次创建服务，但 onStart() 方法会被多次调用。

3. Context.bindService()启动 Service 有关的生命周期方法

onBind()只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，

当调用者与服务已经绑定，多次调用 Context.bindService()方法并不会导致该方法被多次调用。

onUnbind()只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用。

备注：

1. 采用 startService()启动服务

```
Intent intent = new Intent(DemoActivity.this, DemoService.class);
startService(intent);
```

2.Context.bindService()启动

```
Intent intent = new Intent(DemoActivity.this, DemoService.class);
bindService(intent, conn, Context.BIND_AUTO_CREATE);
//unbindService(conn);//解除绑定
```

46. 注册广播有几种方式，这些方式有何优缺点？请谈谈 Android 引入广播机制的用意。

Android 广播机制（两种注册方法）

在 android 下，要想接受广播信息，那么这个广播接收器就得我们自己来实现了，我们可以继承 BroadcastReceiver，就可以有一个广播接受器了。有个接受器还不够，我们还得重写 BroadcastReceiver 里面的 onReceiver 方法，当来广播的时候我们要干什么，这就要我们自己来实现，不过我们可以搞一个信息防火墙。具体的代码：

```
public class SmsBroadCastReceiver extends BroadcastReceiver
{

    @Override
    public void onReceive(Context context, Intent intent)
    {
        Bundle bundle = intent.getExtras();
```

```
            Object[] object = (Object[])bundle.get("pdus");
            SmsMessage sms[]=new SmsMessage[object.length];
            for(int i=0;i<object.length;i++)
            {
                sms[0] = SmsMessage.createFromPdu((byte[])object[i]);
                Toast.makeText(context, "来自"+sms[i].getDisplayOriginatingAddress()+" 的消
息是："+sms[i].getDisplayMessageBody(), Toast.LENGTH_SHORT).show();
            }
            //终止广播，在这里我们可以稍微处理，根据用户输入的号码可以实现短信防火墙。
            abortBroadcast();
        }

}
```

当实现了广播接收器，还要设置广播接收器接收广播信息的类型，这里是信息：android.provider.Telephony.SMS_RECEIVED

我们就可以把广播接收器注册到系统里面，可以让系统知道我们有个广播接收器。这里有两种，一种是代码动态注册：

```
//生成广播处理
smsBroadCastReceiver = new SmsBroadCastReceiver();
//实例化过滤器并设置要过滤的广播
IntentFilter intentFilter = new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
//注册广播
BroadCastReceiverActivity.this.registerReceiver(smsBroadCastReceiver, intentFilter);
```

一种是在 AndroidManifest.xml 中配置广播

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="spl.broadCastReceiver"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".BroadCastReceiverActivity"
```

```
                android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!--广播注册-->
        <receiver android:name=".SmsBroadCastReceiver">
            <intent-filter android:priority="20">
                <action
android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>

    </application>

    <uses-sdk android:minSdkVersion="7" />

    <!-- 权限申请 -->
    <uses-permission
android:name="android.permission.RECEIVE_SMS"></uses-permission>

</manifest>
```

两种注册类型的区别是：

1)第一种不是常驻型广播，也就是说广播跟随程序的生命周期。

2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

47. 请解释下在单线程模型中 Message、Handler、Message Queue、Looper 之间的关

系。

Handler 简介：

一个 Handler 允许你发送和处理 Message 和 Runable 对象，这些对象和一个线程的 MessageQueue 相关联。每一个线程实例和一个单独的线程以及该线程的 MessageQueue 相关联。当你创建一个新的 Handler 时，它就和创建它的线程绑定在一起了。这里，线程我们也可以理解为线程的 MessageQueue。从这一点上来看，Handler 把 Message 和 Runable 对象传递给 MessageQueue，而且在这些对象离开 MessageQueue 时，Handler 负责执行他们。

Handler 有两个主要的用途：（1）确定在将来的某个时间点执行一个或者一些 Message 和 Runnable 对象。（2）在其他线程（不是 Handler 绑定线程）中排入一些要执行的动作。

Scheduling Message，即（1），可以通过以下方法完成：

post(Runnable):Runnable 在 handler 绑定的线程上执行，也就是说不创建新线程。

postAtTime(Runnable,long):
postDelayed(Runnable,long):
sendEmptyMessage(int):
sendMessage(Message):
sendMessageAtTime(Message,long):
sendMessageDelayed(Message,long):

post 这个动作让你把 Runnable 对象排入 MessageQueue,MessageQueue 受到这些消息的时候执行他们，当然以一定的排序。sendMessage 这个动作允许你把 Message 对象排成队列，这些 Message 对象包含一些信息，Handler 的 hanlerMessage(Message)会处理这些 Message.当然，handlerMessage(Message)必须由 Handler 的子类来重写。这是编程人员需要作的事。

当 posting 或者 sending 到一个 Hanler 时，你可以有三种行为：当 MessageQueue 准备好就处

理，定义一个延迟时间，定义一个精确的时间去处理。后两者允许你实现 timeout,tick,和基于时间的行为。

当你的应用创建一个新的进程时，主线程（也就是 UI 线程）自带一个 MessageQueue，这个 MessageQueue 管理顶层的应用对象（像 activities,broadcast receivers 等）和主线程创建的窗体。你可以创建自己的线程，并通过一个 Handler 和主线程进行通信。这和之前一样，通过 post 和 sendmessage 来完成，差别在于在哪一个线程中执行这么方法。在恰当的时候，给定的 Runnable 和 Message 将在 Handler 的 MessageQueue 中被 Scheduled。

Message 简介：

Message 类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给 Handler.Message 对象提供额外的两个 int 域和一个 Object 域，这可以让你在大多数情况下不用作分配的动作。

尽管 Message 的构造函数是 public 的，但是获取 Message 实例的最好方法是调用 Message.obtain(),或者 Handler.obtainMessage()方法，这些方法会从回收对象池中获取一个。

MessageQueue 简介：

这是一个包含 message 列表的底层类。Looper 负责分发这些 message。Messages 并不是直接加到一个 MessageQueue 中，而是通过 MessageQueue.IdleHandler 关联到 Looper。

你可以通过 Looper.myQueue()从当前线程中获取 MessageQueue。

Looper 简介：

Looper 类被用来执行一个线程中的 message 循环。默认情况，没有一个消息循环关联到线程。在线程中调用 prepare()创建一个 Looper，然后用 loop()来处理 messages，直到循环终止。

大多数和 message loop 的交互是通过 Handler。

下面是一个典型的带有 Looper 的线程实现。

```
class LooperThread extends Thread {
    public Handler mHandler;

    public void run() {
        Looper.prepare();

        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };

        Looper.loop();
    }
}
```

48. AIDL 的全称是什么？如何工作？能处理哪些类型的数据？

AIDL 的英文全称是 Android Interface Define Language

当 A 进程要去调用 B 进程中的 service 时，并实现通信，我们通常都是通过 AIDL 来操作的

A 工程：

首先我们在 net.blogjava.mobile.aidlservice 包中创建一个 RemoteService.aidl 文件，在里面我

们自定义一个接口 ,含有方法 get。ADT 插件会在 gen 目录下自动生成一个 RemoteService.java 文件，该类中含有一个名为 RemoteService.stub 的内部类，该内部类中含有 aidl 文件接口的 get 方法。

说明一：aidl 文件的位置不固定，可以任意

然后定义自己的 MyService 类，在 MyService 类中自定义一个内部类去继承 RemoteService.stub 这个内部类，实现 get 方法。在 onBind 方法中返回这个内部类的对象，系统会自动将这个对象封装成 IBinder 对象，传递给他的调用者。

其次需要在 AndroidManifest.xml 文件中配置 MyService 类，代码如下：

```
<!-- 注册服务 -->
<service android:name=".MyService">
    <intent-filter>
    <!-- 指定调用 AIDL 服务的 ID -->
        <action android:name="net.blogjava.mobile.aidlservice.RemoteService" />
     </intent-filter>
</service>
```

为什么要指定调用 AIDL 服务的 ID,就是要告诉外界 MyService 这个类能够被别的进程访问，只要别的进程知道这个 ID，正是有了这个 ID,B 工程才能找到 A 工程实现通信。

说明：AIDL 并不需要权限

B 工程：

首先我们要将 A 工程中生成的 RemoteService.java 文件拷贝到 B 工程中，在 bindService 方法中绑定 aidl 服务

绑定 AIDL 服务就是将 RemoteService 的 ID 作为 intent 的 action 参数。

说明：如果我们单独将 RemoteService.aidl 文件放在一个包里，那个在我们将 gen 目录下的该包拷贝到 B 工程中。如果我们将 RemoteService.aidl 文件和我们的其他类存放在一

起，那么我们在 B 工程中就要建立相应的包，以保证 RmoteService.java 文件的报名正确，我们不能修改 RemoteService.java 文件

bindService(new Inten("net.blogjava.mobile.aidlservice.RemoteService"), serviceConnection, Context.BIND_AUTO_CREATE);

ServiceConnection 的 onServiceConnected(ComponentName name, IBinder service)方法中的 service 参数就是 A 工程中 MyService 类中继承了 RemoteService.stub 类的内部类的对象。

49. 请解释下 Android 程序运行时权限与文件系统权限的区别。

运行时权限 Dalvik(android 授权)

文件系统 linux 内核授权

50. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？请说明原由。

通过直接发送 Uri 把参数带过去，或者通过 manifest 里的 intentfilter 里的 data 属性

51. 你如何评价 Android 系统？优缺点。

答：Android 平台手机 5大优势：

一、开放性

在优势方面，Android 平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

二、挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年 iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些2G 至3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机 IM 软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头 Google 推动的 Android 终端天生就有网络特色，将让用户离互联网更近。

三、丰富的硬件选择

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian 风格手机 一下改用苹果 iPhone ，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移 ,是不是非常方便呢？

四、不受任何限制的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给 Android 难题之一。

五、无缝结合的 Google 应用

如今叱诧互联网的 Google 已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

再说 Android 的5大不足：

一、安全和隐私

由于手机 与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google 这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将

会带来新一轮的隐私危机。

二、首先开卖 Android 手机的不是最大运营商

众所周知，T-Mobile 在23日，于美国纽约发布 了 Android 首款手机 G1。但是在北美市场，最大的两家运营商乃 AT&T 和 Verizon ,而目前所知取得 Android 手机销售权的仅有 T-Mobile 和 Sprint，其中 T-Mobile 的3G 网络相对于其他三家也要逊色不少，因此，用户可以买账购买 G1，能否体验到最佳的3G 网络服务则要另当别论了！

三、运营商仍然能够影响到 Android 手机

在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android 手机的另一发售运营商 Sprint 就将在其机型中内置其手机商店程序。

四、同类机型用户减少

在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于 Android 平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。

五、过分依赖开发商缺少标准配置

在使用 PC 端的 Windows Xp 系统的时候，都会内置微软 Windows Media Player 这样一个浏览器程序，用户可以选择更多样的播放器，如 Realplay 或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android 平台中，由于其开放性，软件更多依赖第三方厂商，比如 Android 系统的 SDK 中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统一性。

52. 什么是 ANR 如何避免它?

答：ANR：Application Not Responding，五秒

在 Android 中，活动管理器和窗口管理器这两个系统服务负责监视应用程序的响应。当出现下列情况时，Android 就会显示 ANR 对话框了：

对输入事件(如按键、触摸屏事件)的响应超过5秒

意向接受器(intentReceiver)超过10秒钟仍未执行完毕

Android 应用程序完全运行在一个独立的线程中(例如 main)。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发 ANR。因为此时，你的应用程序已经没有机会去响应输入事件和意向广播(Intent broadcast)。

因此，任何运行在主线程中的方法，都要尽可能的只做少量的工作。特别是活动生命周期中的重要方法如 onCreate()和 onResume()等更应如此。潜在的比较耗时的操作，如访问网络和数据库;或者是开销很大的计算，比如改变位图的大小，需要在一个单独的子线程中完成(或者是使用异步请求，如数据库操作)。但这并不意味着你的主线程需要进入阻塞状态已等待子线程结束 -- 也不需要调用 Therad.wait()或者 Thread.sleep()方法。取而代之的是，主线程为子线程提供一个句柄(Handler)，让子线程在即将结束的时候调用它(xing:可以参看 Snake 的例子，这种方法与以前我们所接触的有所不同)。使用这种方法涉及你的应用程序，能够保证你的程序对输入保持良好的响应，从而避免因为输入事件超过5秒钟不被处理而产生的 ANR。这种实践需要应用到所有显示用户界面的线程，因为他们都面临着同样的超时问题。

53. 什么情况会导致 Force Close ?如何避免?能否捕获导致其的异常?

答：一般像空指针啊，可以看起 logcat，然后对应到程序中 来解决错误

54. Android本身的api并未声明会抛出异常，则其在运行时有无可能抛出runtime异常，你遇到过吗?诺有的话会导致什么问题?如何解决?

55. 简要解释一下activity、 intent 、intent filter、service、Broadcase、BroadcaseReceiver

答：一个activity呈现了一个用户可以操作的可视化用户界面

一个service不包含可见的用户界面，而是在后台无限地运行

可以连接到一个正在运行的服务中，连接后，可以通过服务中暴露出来的借口与其进行通信

一个broadcast receiver是一个接收广播消息并作出回应的component，broadcast receiver没有界面

intent:content provider在接收到ContentResolver的请求时被激活。

activity, service和broadcast receiver是被称为intents的异步消息激活的。

一个intent是一个Intent对象，它保存了消息的内容。对于activity和service来说，它指定了请求的操作名称和待操作数据的URI

Intent对象可以显式的指定一个目标component。如果这样的话，android会找到这个component(基于manifest文件中的声明)并激活它。但如果一个目标不是显式指定的，android必须找到响应intent的最佳component。

它是通过将Intent对象和目标的intent filter相比较来完成这一工作的。一个component

的 intent filter 告诉 android 该 component 能处理的 intent。intent filter 也是在 manifest 文件中声明的。

56. IntentService 有何优点?

答：IntentService 的好处

* Acitivity 的进程，当处理 Intent 的时候，会产生一个对应的 Service

* Android 的进程处理器现在会尽可能的不 kill 掉你

* 非常容易使用

57. 横竖屏切换时候 activity 的生命周期?

1、不设置 Activity 的 android:configChanges 时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次

2、设置 Activity 的 android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次

3、设置 Activity 的 android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行 onConfigurationChanged 方法

如何将 SQLite 数据库(dictionary.db 文件)与 apk 文件一起发布?

解答：可以将 dictionary.db 文件复制到 Eclipse Android 工程中的 res aw 目录中。所有在 res aw 目录中的文件不会被压缩，这样可以直接提取该目录中的文件。可以将 dictionary.db 文件复制到 res aw 目录中

58. 如何将打开 res aw 目录中的数据库文件?

解答：在 Android 中不能直接打开 res aw 目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或 SD 卡的某个目录中，然后再打开该数据库文件。复制的

基本方法是使用 getResources().openRawResource 方法获得 res aw 目录中资源的 InputStream 对象，然后将该 InputStream 对象中的数据写入其他的目录中相应文件中。在 Android SDK 中可以使用 SQLiteDatabase.openOrCreateDatabase 方法来打开任意目录中的 SQLite 数据库文件。

59. Android 引入广播机制的用意?

答：a:从 MVC 的角度考虑(应用程序内)

其实回答这个问题的时候还可以这样问，android 为什么要有那4大组件，现在的移动开发模型基本上也是照搬的 web 那一套 MVC 架构，只不过是改了点嫁妆而已。android 的四大组件本质上就是为了实现移动或者说嵌入式设备上的 MVC 架构，它们之间有时候是一种相互依存的关系，有时候又是一种补充关系，引入广播机制可以方便几大组件的信息和数据交互。

b：程序间互通消息(例如在自己的应用程序内监听系统来电)

c：效率上(参考 UDP 的广播协议在局域网的方便性)

d：设计模式上(反转控制的一种应用，类似监听者模式)

60. Android dvm 的进程和 Linux 的进程，应用程序的进程是否为同一个概念

DVM 指 dalivk 的虚拟机。每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。而每一个 DVM 都是在 Linux 中的一个进程，所以说可以认为是同一个概念。

61. sim 卡的 EF 文件有何作用

sim 卡的文件系统有自己规范，主要是为了和手机通讯，sim 本 身可以有自己的操作系统，EF 就是作存储并和手机通讯用的

62. 嵌入式操作系统内存管理有哪几种， 各有何特性

页式，段式，段页，用到了 MMU,虚拟空间等技术

63. 什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗?

嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种， 而 android 是基于 linux 内核的，因此属于软实时。

64. 一条最长的短信息约占多少 byte?

中文70(包括标点)，英文160，160个字节。

65. android 中的动画有哪几类，它们的特点和区别是什么?

两种，一种是 Tween 动画、还有一种是 Frame 动画。Tween 动画，这种实现方式可以使视图组件移动、放大、缩小以及产生透明度的变化;另一种 Frame 动画，传统的动画方法，通过顺序的播放排列好的图片来实现，类似电影。

66. handler 机制的原理

andriod 提供了 Handler 和 Looper 来满足线程间的通信。Handler 先进先出原则。Looper 类用来管理特定线程内对象之间的消息交换(Message Exchange)。

1)Looper: 一个线程可以产生一个 Looper 对象，由它来管理此线程里的 Message Queue(消息队列)。

2)Handler: 你可以构造 Handler 对象来与 Looper 沟通，以便 push 新消息到 Message Queue 里;或者接收 Looper 从 Message Queue 取出)所送来的消息。

3) Message Queue(消息队列):用来存放线程放入的消息。

4)线程：UI thread 通常就是 main thread，而 Android 启动程序时会替它建立一个 Message Queue。

67. 说说 mvc 模式的原理，它在 android 中的运用

MVC(Model_view_contraller)"模型_视图_控制器"。 MVC 应用程序总是由这三个部分组成。Event(事件)导致 Controller 改变 Model 或 View，或者同时改变两者。只要 Controller 改变了 Models 的数据或者属性，所有依赖的 View 都会自动更新。类似的，只要 Contro

68. DDMS 和 TraceView 的区别?

DDMS 是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView 是程序性能分析器 。

69. java 中如何引用本地语言

可以用 JNI（java native interface java 本地接口）接口 。

70. 谈谈 Android 的 IPC（进程间通信）机制

IPC 是内部进程通信的简称， 是共享"命名管道"的资源。Android 中的 IPC 机制是为了让 Activity 和 Service 之间可以随时的进行交互，故在 Android 中该机制，只适用于 Activity 和

Service 之间的通信，类似于远程方法调用，类似于 C/S 模式的访问。通过定义 AIDL 接口文件来定义 IPC 接口。Servier 端实现 IPC 接口，Client 端调用 IPC 接口本地代理。

71. NDK 是什么

NDK 是一些列工具的集合，NDK 提供了一系列的工具，帮助开发者迅速的开发 C/C++的动态库，并能自动将 so 和 java 应用打成 apk 包。

NDK 集成了交叉编译器，并提供了相应的 mk 文件和隔离 cpu、平台等的差异，开发人员只需简单的修改 mk 文件就可以创建出 so

第二章 Android 面试题总结加强版（一）

1.activity 的生命周期。

方法	描述	可被杀死	下一个
onCreate()	在 activity 第一次被创建的时候调用。这里是你做所有初始化设置的地方——创建视图、设置布局、绑定数据至列表等。如果曾经有状态记录（参阅后述 Saving Activity State。），则调用此方法时会传入一个包含着此 activity 以前状态的包对象做为参数。 总继之以 onStart()。	否	onStart()
onRestart()	在 activity 停止后 onStop()，在再次启动之前被调用。 总继之以 onStart()。	否	onStart()

onStart()	当 activity 正要变得为用户所见时被调用。 当 activity 转向前台时继以 onResume()，在 activity 变为隐藏时继以 onStop()。	否	onResume() or onStop()
onResume()	在 activity 开始与用户进行交互之前被调用。此时 activity 位于堆栈顶部，并接受用户输入。 继之以 onPause()。	否	onPause()
onPause()	当系统将要启动另一个 activity 时调用。此方法主要用来将未保存的变化进行持久化，停止类似动画这样耗费 CPU 的动作等。这一切动作应该在短时间内完成，因为下一个 activity 必须等到此方法返回后才会继续。 当 activity 重新回到前台是继以 onResume()。当 activity 变为用户不可见时继以 onStop()。	是	onResume() or onStop()
onStop()	当 activity 不再为用户可见时调用此方法。这可能发生在它被销毁或者另一个 activity（可能是现存的或者是新的）回到运行状态并覆盖了它。 如果 activity 再次回到前台跟用户交互则继以 onRestart()，如果关闭 activity 则继以 onDestroy()。	是	onRestart() or onDestroy()

onDestroy()	在 activity 销毁前调用。这是 activity 接收的最后一个调用。这可能发生在 activity 结束（调用了它的 finish() 方法）或者因为系统需要空间所以临时的销毁了此 acitivity 的实例时。你可以用 isFinishing() 方法来区分这两种情况。	是	nothing

（补充）

1、onCreate():当 Activity 被创建的时候调用（第一次）。操作：设置布局文件，初始化视图，绑定数据文件等。

2、onStart():当 Activity 能被我们看到的时候。

3、onResume():当 Activity 获得用户的焦点的时候，就是能被用户操作的时候。

4、onPause()[pause 暂停的意思]:Activity 暂停。应用程序启动了另一个 Activity 的时候。例子：来了一个电话，系统启动了电话 Activity。在这个函数里要做的就是把 Activity 的数据保存起来，当接完电话的时候，再把这些数据读出来，把原来的 Activity 还原出来。

5、onstop()：当第二个 Activity 把第一个 Activity 完全遮挡住了的时候。对话框并没有把原来的 Activity 完全遮挡起来，不会调用。

6、onDestroy():销毁 Activity。1）调用了 finish()方法。2）系统资源不够用了。

函数调用过程：

启动第一个 Activity 的时候：

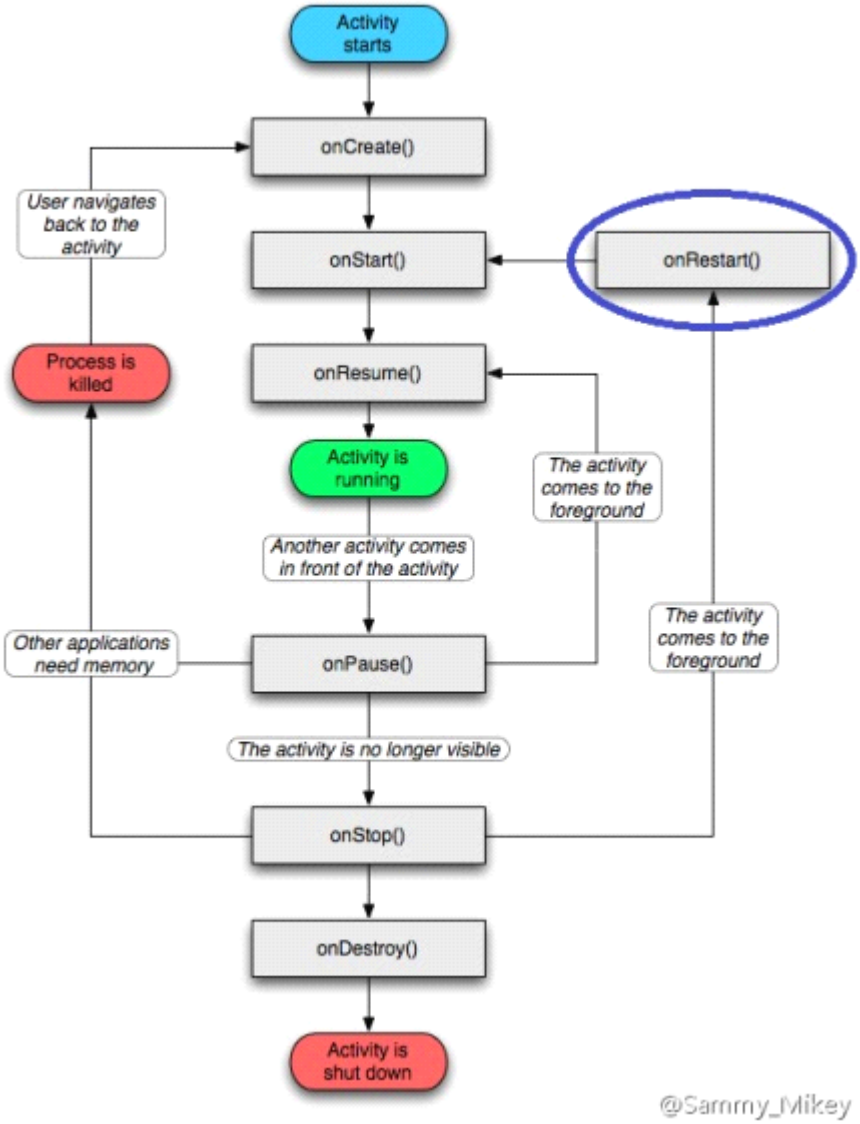
第一次创建 onCreate()-->Activity 可见了 onStart()-->Activity 可以操作了 onResume()。

点击第一个 Activity 上的按钮通过 Intent 跳到第二个 Activity：

第一个 Activity 暂停 onPause()-->创建第二个 ActivityonCreate()-->Activity 可见 onStart()-->Activity 可操作 onResume()-->第一个 Activity 被第二个 Activity 完全遮盖 onStop()(如果调用了 finish(),或者系统资源紧缺，则会被销毁 onDestory())。

点击系统返回功能建，从第二个 Activity 回到第一个 Activity ：

第二个 Activity 暂停 onPause()-->第一个 Activity 重启动 OnRestart()(并没有被销毁,如果销毁了则要创建 onCreate())-->第一个 Activity 可见 onStart()-->第一个 Activity 可操作 onResume()-->第二个 Activity 被完全遮盖 onStop()(如果调用了 finish(),或者系统资源紧缺，则会被销毁 onDestory())。



2.横竖屏切换时候 activity 的生命周期

1.不设置 Activity 的 android:configChanges 时,切屏会重新调用各个生命周期,切横屏时会执行一次,切竖屏时会执行两次.

2.设置 Activity 的 android:configChanges="orientation"时,切屏还是会重新调用各个生命周期,切横、竖屏时只会执行一次.

3.设置 Activity 的 android:configChanges="orientation|keyboardHidden"时,切屏不会重新调用各个生命周期,只会执行 onConfigurationChanged 方法.

3.android 中的动画有哪几类，它们的特点和区别是什么？

Android 提供两种创建简单动画的机制：tweened animation（补间动画） 和 frame-by-frame animation（帧动画）.

- tweened animation：通过对场景里的对象不断做图像变换(平移、缩放、旋转)产生动画效果
- frame-by-frame animation：顺序播放事先做好的图像，跟电影类似

这两种动画类型都能在任何 View 对象中使用，用来提供简单的旋转计时器，activity 图标及其他有用的 UI 元素。Tweened animation 被 andorid.view.animation 包所操作；frame-by-frame animation 被 android.graphics.drawable.AnimationDrawable 类所操作。想了解更多关于创建 tweened 和 frame-by-frame 动画的信息，读一下 Dev Guide-Graphics-2D Graphics 里面相关部分的讨论。

Animation 是以 XML 格式定义的，定义好的 XML 文件存放在 res/anim 中。由于 Tween Animation 与 Frame-by-frame Animation 的定义、使用都有很大的差异，我们将分开介绍，本篇幅中主要介绍 Tween Animation 的定义与使用。按照 XML 文档的结构【父节点，子节点，属性】来介绍 Tween Animation，其由4种类型：

- Alpha： 渐变透明度动画效果
- Scale：渐变尺寸伸缩动画效果
- Translate：画面转换位置移动动画效果
- Rotate： 画面转换角度移动动画效果

在介绍以上4种 类型前，先介绍 Tween Animation 共同的节点属性。

表一		
属性[类型]	功能	

Duration[long]	属性为动画持续时间	时间以毫秒为单位
fillAfter [boolean]	当设置为 true ，该动画转化在动画结束后被应用	
fillBefore[boolean]	当设置为 true ，该动画转化在动画开始前被应用	
interpolator	指定一个动画的插入器	有一些常见的插入器 accelerate_decelerate_interpolator 加速-减速 动画插入器 accelerate_interpolator 加速-动画插入器 decelerate_interpolator 减速- 动画插入器 其他的属于特定的动画效果
repeatCount[int]	动画的重复次数	
RepeatMode[int]	定义重复的行为	1：重新开始 2：plays backward
startOffset[long]	动画之间的时间间隔，从上次动画停多少时间开始执行下个动画	
zAdjustment[int]	定义动画的 Z Order 的改变	0：保持 Z Order 不变 1：保持在最上层 -1：保持在最下层

下面我们开始结 合具体的例子，分别介绍4种类型各自特有的节点元素。

表二

XML 节点	功能说明
alpha	渐变透明度动画效果
<alpha android:fromAlpha=”0.1″ android:toAlpha=”1.0″ android:duration=”3000″ />	
fromAlpha 属性为动画起始时透明度	0.0表示完全不透明
toAlpha 属性为动画结束时透明度	1.0表示完全透明 以上值取0.0-1.0之间的 float 数据类型的数字

表三

scale	渐变尺寸伸缩动画效果
<scale android:interpolator= “@android:anim/accelerate decelerate interpolator” android:fromXScale=”0.0″	

android:toXScale=”1.4″ android:fromYScale=”0.0″ android:toYScale=”1.4″ android:pivotX=”50%” android:pivotY=”50%” android:fillAfter=”false” android:startOffset=“700” android:duration=”700″ android:repeatCount=”10″ />		
fromXScale[float] fromYScale[float]	为动画起始时，X、Y 坐标上的伸缩尺寸	0.0表示收缩到没有 1.0表示正常无伸缩 值小于1.0表示收缩 值大于1.0表示放大
toXScale[float] toYScale[float]	为动画结束时，X、Y 坐标上的伸缩尺寸	
pivotX[float] pivotY[float]	为动画相对于物件的 X、Y 坐标的开始位置	属性值说明：从0%-100%中取值，50%为物件的 X 或 Y 方向坐标上的中点位置

表四		
translate	画面转换位置移动动画效果	
<translate android:fromXDelta=”30″ android:toXDelta=”-80″ android:fromYDelta=”30″ android:toYDelta=”300″ android:duration=”2000″ />		
fromXDelta toXDelta	为动画、结束起始时 X 坐标上的位置	
fromYDelta	为动画、结束起始时 Y 坐标上的位置	

toYDelta			

表五

rotate		画面转移旋转动画效果
<rotate android:interpolator=”@android:anim/accelerate_decelerate_interpolator” android:fromDegrees=”0″ android:toDegrees=”+350″ android:pivotX=”50%” android:pivotY=”50%” android:duration=”3000″ />		
fromDegrees	为动画起始时物件的角度	说明 当角度为负数——表示逆时针旋转 当角度为正数——表示顺时针旋转 (负数 from——to 正数:顺时针旋转) (负数 from——to 负数:逆时针旋转) (正数 from——to 正数:顺时针旋转) (正数 from——to 负数:逆时针旋转)
toDegrees	为动画结束时物 件旋转的角度 可以大于360度	
pivotX pivotY	为动画相对于物件的 X、Y 坐标的开始位置	说明：以上两个属性值 从0%-100%中取值 50%为物件的 X 或 Y 方向坐标上的中点位置

Android SDK 提供了基类：Animation，包含大量的 set/getXXXX()函数来设置、读取 Animation 的属性，也就是前面表一中显示的各种属性。Tween Animation 由4种类型：alpha、scale、translate、roate，在 Android SDK 中提供了相应的类，Animation 类派生出了 AlphaAnimation、ScaleAnimation、TranslateAnimation、RotateAnimation 分别实现了 改变 Alpha 值 、 伸缩 、 平移 、 旋转 等动画，每个子类都在父类的基础上增加了各自独有的属性。

（补充）

Android 的动画效果分为两种，一种是 tweenedanimation(补间动画)，第二种是 frame by frame animation。一般我们用的是第一种。补间动画又分为AlphaAnimation,透明度转换 RotateAnimation,旋转转换 ScaleAnimation,缩放转换 TranslateAnimation 位置转换（移动）。

动画效果在 anim 目录下的 xml 文件中定义，在程序中用AnimationUtils.loadAnimation(Context

context,int ResourcesId)载入成 Animation 对象，在需要显示动画效果时，执行需要动画的 View 的 startAnimation 方法，传入 Animation,即可。切换 Activity 也可以应用动画效果，在 startActivity 方法后，执行 overridePendingTransition 方法，两个参数分别是切换前的动画效果，切换后的动画效果

4. 一条最长的短信息约占多少 byte?

中文70(包括标点)，英文160个字节。

5.handler 机制的原理

andriod 提供了 Handler 和 Looper 来满足线程间的通信。Handler 先进先出原则。Looper 类用来管理特定线程内对象之间的消息交换(Message Exchange)。

1)Looper: 一个线程可以产生一个 Looper 对象，由它来管理此线程里的 Message Queue(消息队列)。

2)Handler: 你可以构造 Handler 对象来与 Looper 沟通，以便 push 新消息到 Message Queue 里;或者接收 Looper 从 Message Queue 取出)所送来的消息。

3) Message Queue(消息队列):用来存放线程放入的消息。

4)线程：UI thread 通常就是 main thread，而 Android 启动程序时会替它建立一个 Message Queue。

6.什么是嵌入式实时操作系统，Android 操作系统属于实时操作系统吗?

嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而 android 是基于 linux 内核的，因此属于软实时

7.android 中线程与线程，进程与进程之间如何通信

1、一个 Android 程序开始运行时，会单独启动一个 Process。

默认情况下，所有这个程序中的 Activity 或者 Service 都会跑在这个 Process。

默认情况下，一个 Android 程序也只有一个 Process，但一个 Process 下却可以有许多个 Thread。

2、一个 Android 程序开始运行时，就有一个主线程 Main Thread 被创建。该线程主要负责 UI 界面的显示、更新和控件交互，所以又叫 UI Thread。

一个 Android 程序创建之初，一个 Process 呈现的是单线程模型--即 Main Thread，所有的任务都在一个线程中运行。所以，Main Thread 所调用的每一个函数，其耗时应该越短越好。而对于比较费时的工作，应该设法交给子线程去做，以避免阻塞主线程（主线程被阻塞，会导致程序假死 现象）。

3、Android 单线程模型：Android UI 操作并不是线程安全的并且这些操作必须在 UI 线程中执行。如果在子线程中直接修改 UI，会导致异常。

8.Android dvm 的进程和 Linux 的进程，应用程序的进程是否为同一个概念

DVM 指 dalivk 的虚拟机。每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。而每一个 DVM 都是在 Linux 中的一个进程，所以说可以认为是同一个概念。

9.sim 卡的 EF 文件有何作用

sim 卡的文件系统有自己规范，主要是为了和手机通讯，sim 本 身可以有自己的操作系统，EF 就是作存储并和手机通讯用的

11.让 Activity 变成一个窗口：Activity 属性设定

讲点轻松的吧,可能有人希望做出来的应用程序是一个漂浮在手机主界面的东西，那么很 简单你只需要设置 一下 Activity 的主题就可以了在 AndroidManifest.xml 中定义 Activity 的 地方一句话：

Xml 代码

```
1. android :theme="@android:style/Theme.Dialog"
```

这就使你的应用程序变成对话框的形式弹出来了，或者

Xml 代码

```
1. android:theme="@android:style/Theme.Translucent"
```

就变成半透明的,[友情提示-.-]类似的这种 activity 的属性可以在 android.R.styleable 类的 AndroidManifestActivity 方法中看到，AndroidManifest.xml 中所有元素的属性的介绍都可以参考这个类 android.R.styleable

上面说的是属性名称，具体有什么值是在 android.R.style 中 可以看到，比如这个

"@android:style/Theme.Dialog" 就对应于 android.R.style.Theme_Dialog ,('_'换成'.' < --注意：这个是文章内容不是笑脸)就可以用在描述文件 中了,找找类定义和描述文件中的对应关系就都明白了。

12.如何将 SQLite 数据库(dictionary.db 文件)与 apk 文件一起发布?

解答：可以将 dictionary.db 文件复制到 Eclipse Android 工程中的 res raw 目录中。所有在 res raw 目录中的文件不会被压缩，这样可以直接提取该目录中的文件。可以将 dictionary.db 文件复制到 res raw 目录中

13.如何将打开 res aw 目录中的数据库文件?

解答：在 Android 中不能直接打开 res raw 目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或 SD 卡的某个目录中，然后再打开该数据库文件。复制的基本方法是使用 getResources().openRawResource 方法获得 res raw 目录中资源的 InputStream 对象，然后将该 InputStream 对象中的数据写入其他的目录中相应文件中。在 Android SDK 中可以使用 SQLiteDatabase.openOrCreateDatabase 方法来打开任意目录中的 SQLite 数据库文件。

14.在 android 中 mvc 的具体体现

Android 的官方建议应用程序的开发采用 MVC 模式。何谓 MVC？先看看下图

MVC 是 Model,View,Controller 的缩写，从上图可以看出 MVC 包含三个部分：

.. 模型（Model）对象：是应用程序的主体部分，所有的业务逻辑都应该写在该层。

.. 视图（View）对象：是应用程序中负责生成用户界面的部分。也是在整个 MVC 架构中用户唯一可以看到的一层，接收用户的输入，显示处理结果。

.. 控制器（Control）对象：是根据用户的输入，控制用户界面数据显示及更新 Model 对象状态的部分，控制器更重要的一种导航功能，响应用户出发的相关事件，交给 M 层处理。

Android 鼓励弱耦合和组件的重用，在 Android 中 MVC 的具体体现如下

1)视图层（view）：一般采用 xml 文件进行界面的描述，使用的时候可以非常方便的引入，当然，如何你对 android 了解的比较多的话，就一定可以想到在 android 中也可以使用 javascript+html 等的方式作为 view 层，当然这里需要进行 java 和 javascript 之间的通信，幸运的是，android 提供了它们之间非常方便的通信实现。

2)控制层（controller）：android 的控制层的重任通常落在了众多的 acitvity 的肩上，这句话也就暗含了不要在 acitivity 中写代码，要通过 activity 交给 model 层做业务逻辑的处理， 这样做的另外一个原因是 android 中的 acitivity 的响应时间是5s，如果耗时的操作放在这里，程序就很容易被回收掉。

3)模型层（model）：对数据库的操作、对网络等的操作都应该在 model 里面处理，当然对业务计算等操作也是必须放在的该层的。

15.Android 系统的架构

android 的系统架构和其操作系统一样，采用了分层的架构。从架构图看，android 分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和 linux 核心层。

1.应用程序

Android 会同一系列核心应用程序包一起发布，该应用程序包包括 email 客户端，SMS 短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用 JAVA 语言编写的。

2.应用程序框架

开发人员也可以完全访问核心应用程序所使用的 API 框架。该应用程序的架构设计简化了组件的重用;任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块(不过得遵循框架的安全性限制)。同样，该应用程序重用机制也使用户可以方便的替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统, 其中包括;

* 丰富而又可扩展的视图(Views)，可以用来构建应用程序， 它包括列表(lists)，网格(grids)，文本框(text boxes)，按钮(buttons)， 甚至可嵌入的 web 浏览器。

* 内容提供器(Content Providers)使得应用程序可以访问另一个应用程序的数据(如联系人数据库)， 或者共享它们自己的数据

* 资源管理器(Resource Manager)提供 非代码资源的访问，如本地字符串，图形，和布局文件(layout files)。

* 通知管理器 (Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。

* 活动管理器(Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。

有关更多的细节和怎样从头写一个应用程序，请参考 如何编写一个 Android 应用程序.

3.系统运行库

1)程序库

Android 包含一些 C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库：

* 系统 C 库 - 一个从 BSD 继承来的标准 C 系统函数库(libc)， 它是专门为基于 embedded linux 的设备定制的。

* 媒体库 - 基于 PacketVideo OpenCORE;该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG 。

* Surface Manager - 对显示子系统的管理，并且为多个应用程序提 供了2D 和3D 图层的无缝融合。

* LibWebCore - 一个最新的 web 浏览器引擎用，支持 Android 浏览器和一个可嵌入的 web 视图。

* SGL - 底层的2D 图形引擎

* 3D libraries - 基于 OpenGL ES 1.0 APIs 实现;该库可以使用硬件 3D 加速(如果可用)或者使用高度优化的3D 软加速。

* FreeType -位图(bitmap)和矢量(vector)字体显示。

* SQLite - 一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

2)Android 运行库

Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。

每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。 Dalvik 虚拟机执行(.dex)的 Dalvik 可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由 JAVA 编译器编译，然后通过 SDK 中 的 “dx” 工具转化成.dex 格式由虚拟机执行。

Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

4.Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。

（补充）

android 系统架构分从下往上为 linux 内核层、运行库、应用程序框架层、和应用程序层。

linuxkernel：负责硬件的驱动程序、网络、电源、系统安全以及内存管理等功能。

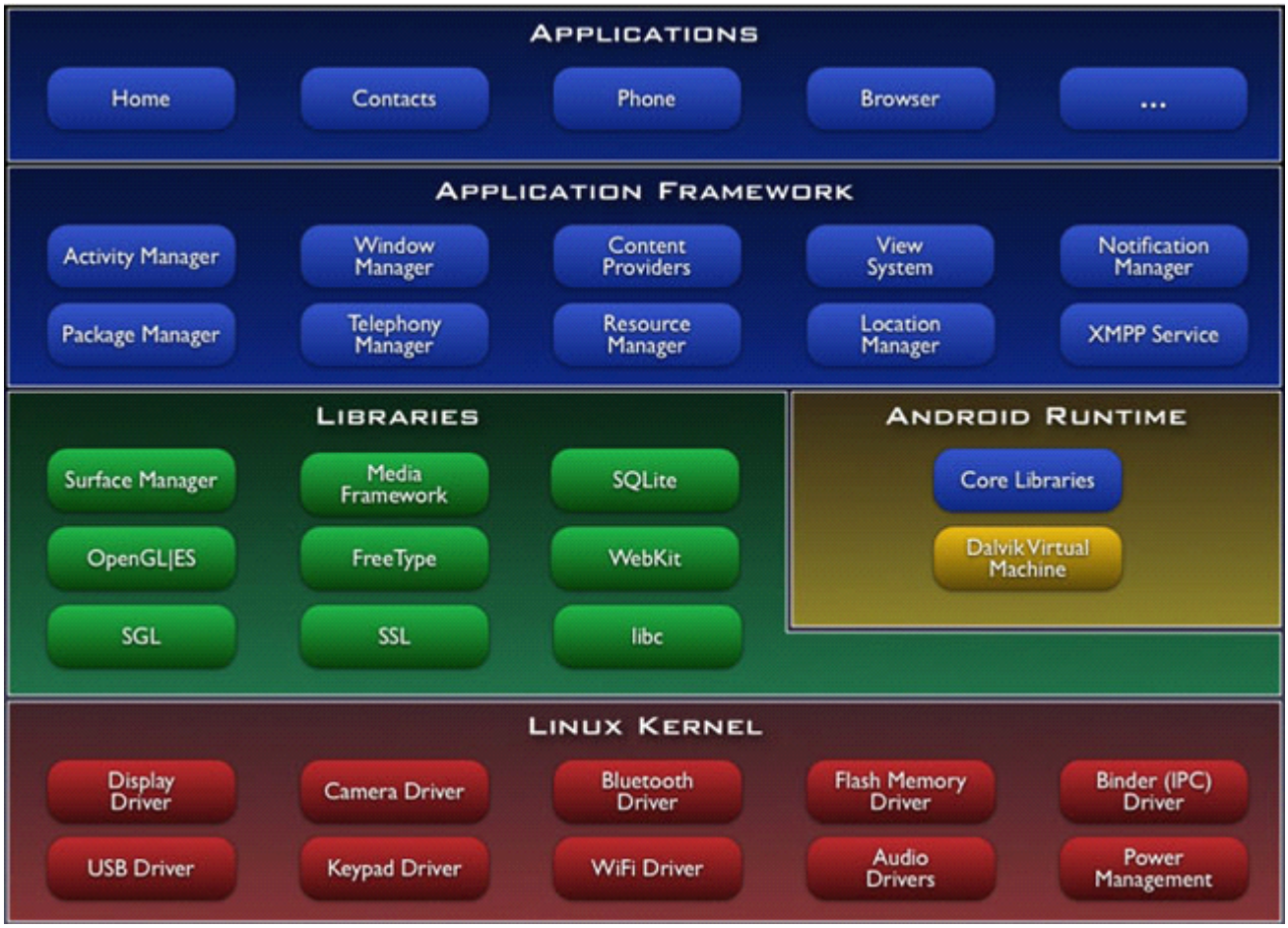
libraries 和 android runtime：libraries：即 c/c++函数库部分，大多数都是开放源代码的函数库，例如 webkit（引擎），该函数库负责 android 网页浏览器的运行，例如标准的 c 函数库 libc、openssl、sqlite 等，当然也包括支持游戏开发 2dsgl 和 3dopengles，在多媒体方面有 mediaframework 框架来支持各种影音和图形文件的播放与显示，例如 mpeg4、h.264、mp3、 aac、amr、jpg 和 png 等众多的多媒体文件格式。android 的 runtime 负责解释和执行生成的 dalvik 格式的字节码。

applicationframework（应用软件架构），java 应用程序开发人员主要是使用该层封装好的 api 进行快速开发。

applications:该层是 java 的应用程序层，android 内置的 googlemaps、e-mail、即时通信工具、浏览器、mp3 播放器等处于该层，java 开发人员开发的程序也处于该层，而且和内置的应用程序具有平等的位置，可以调用内置的应用程序，也可以替换内置的应用程序。

上面的四个层次，下层为上层服务，上层需要下层的支持，调用下层的服务，这种严格分层的方式带来的极大的稳定性、灵活性和可扩展性，使得不同层的开发人员可以按照规范专心特定层的开发。

android 应用程序使用框架的 api 并在框架下运行，这就带来了程序开发的高度一致性，另一方面也告诉我们，要想写出优质高效的程序就必须对整个 applicationframework 进行非常深入的理解。精通 applicationframework，你就可以真正的理解 android 的设计和运行机制，也就更能够驾驭整个应用层的开发。



第三章 Android 面试题总结加强版（二）

16.Android 常用控件的信息

单选框(RadioButton 与 RadioGroup)：

RadioGroup 用于对单选框进行分组，相同组内的单选框只有一个单选框被选中。

事件：setOnCheckedChangeListener()，处理单选框被选择事件。把

RadioGroup.OnCheckedChangeListener 实例作为参数传入。

多选框(CheckBox):

每个多选框都是独立的，可以通过迭代所有的多选框，然后根据其状态是否被选中在获取其值。

事件：setOnCheckedChangeListener()，处理多选框被选择事件。把

CheckBox.OnCheckedChangeListener()实例作为参数传入。

下拉列表框(Spinner)：

Spinner.getItemAtPosition(Spinner.getSelectedItemPosition());获取下拉列表框的值。

事件：setOnItemSelectedListener(),处理下拉列表框被选择事件把

Spinner.OnItemSelectedListener()实例作为参数传入。

拖动条(SeekBar)：

SeekBar.getProgress()获取拖动条当前值

事件:setOnSeekBarChangeListener()，处理拖动条值变化事件，把

SeekBar.OnSeekBarChangeListener 实例作为参数传入。

菜单(Menu):

重写 Activity 的 onCreatOptionMenu(Menu menu)方法，该方法用于创建选项菜单，当

用户按下手机的"Menu"按钮时就会显示创建好的菜单，在 onCreatOptionMenu(Menu

Menu)方法内部可以调用 Menu.add()方法实现菜单的添加。

重写 Activity 的 onMenuItemSelected()方法，该方法用于处理菜单被选择事件。

进度对话框(ProgressDialog)：

创建并显示一个进度对话框：ProgressDialog.show(ProgressDialogActivity.this,"请稍等

"，"数据正在加载中....."，true)；

设置对话框的风格：setProgressStyle()

ProgressDialog.STYLE_SPINNER　旋转进度条风格(为默认风格)

ProgressDialog.STYLE_HORIZONTAL 横向进度条风格

（补充）

下面是各种常用控件的事件监听的使用

①EditText（编辑框）的事件监听--—OnKeyListener

②RadioGroup、RadioButton（单选按钮）的事件监听——OnCheckedChangeListener

③CheckBox（多选按钮）的事件监听---OnCheckedChangeListener

④Spinner（下拉列表）的事件监听---OnItemSelectedListener

⑤Menu（菜单）的事件处理——onMenuItemSelected

⑥Dialog（对话框）的事件监听——DialogInterface.OnClickListener()

http://www.iteye.com/topic/1060815

17.请介绍下 Android 中常用的五种布局

Android 布局是应用界面开发的重要一环，在 Android 中，共有五种布局方式，分别是：

FrameLayout（帧布局），LinearLayout （线性布局），

AbsoluteLayout（绝对布局），RelativeLayout（相对布局），TableLayout（表格布局）。

1.FrameLayout

这个布局可以看成是墙脚堆东西，有一个四方的矩形的左上角墙脚，我们放了第一个东西，要再放一个，那就在放在原来放的位置的上面，这样依次的放，会盖住原来的东西。这个布局比较简单，也只能放一点比较简单的东西。

2.LinearLayout

线性布局，这个东西，从外框上可以理解为一个 div，他首先是一个一个从上往下罗列在屏幕上。每一个 LinearLayout 里面又可分为垂直布局（android:orientation="vertical"）和水平布局（android:orientation="horizontal" ）。当垂直布局时，每一行就只有一个元素，多个元素依次垂直往下；水平布局时，只有一行，每一个元素依次向右排列。

LinearLayout 中有一个重要的属性 android:layout_weight="1"，这个 weight 在垂直布局时，代表行距；水平的时候代表列宽；weight 值越大就越大。

3.AbsoluteLayout

绝对布局犹如 div 指定了 absolute 属性，用 X,Y 坐标来指定元素的位置 android:layout_x="20px" android:layout_y="12px" 这种布局方式也比较简单，但是在

垂直随便切换时，往往会出问题，而且多个元素的时候，计算比较麻烦。

4.RelativeLayout

相对布局可以理解为某一个元素为参照物，来定位的布局方式。主要属性有：

相对于某一个元素

android:layout_below="@id/aaa" 该元素在 id 为 aaa 的下面

android:layout_toLeftOf="@id/bbb" 该元素在 id 为 bbb 的左边

相对于父元素的地方

android:layout_alignParentLeft="true" 与父元素左对齐

android:layout_alignParentRight="true" 与父元素右对齐

还可以指定边距等，具体详见 API

5.TableLayout

表格布局类似 Html 里面的 Table。每一个 TableLayout 里面有表格行 TableRow，TableRow 里面可以具体定义每一个元素，设定他的对齐方式 android:gravity="" 。

每一个布局都有自己适合的方式，另外，这五个布局元素可以相互嵌套应用，做出美观的界面。

18.如何启用 Service，如何停用 Service

Android 中的服务和 windows 中的服务是类似的东西，服务一般没有用户操作界面，它运行于系统中不容易被用户发觉，可以使用它开发如监控之类的程序。服务的开发比较简单，如下：

第一步：继承 Service 类

```
public class SMSService extends Service {
}
```

第二步：在 AndroidManifest.xml 文件中的<application>节点里对服务进行配置:

```
<service android:name=".SMSService" />
```

服务不能自己运行，需要通过调用 Context.startService()或 Context.bindService()方法启动服务。这两个方法都可以启动 Service，但是它们的使用场合有所不同。使用 startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。使用 bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止，大有“不求同时生，必须同时死”的特点。

如果打算采用 Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的 onCreate()方法，接着调用 onStart()方法。如果调用 startService()方法前服务已经被创建，多次调用 startService()方法并不会导致多次创建服务，但会导致多次调用 onStart()方法。采用 startService()方法启动的服务，只能调用 Context.stopService()方法结束服务，服务结束时会调用 onDestroy()方法。

如果打算采用 Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的 onCreate()方法，接着调用 onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的 onUnbind()方法，接着调用 onDestroy()方法。如果调用 bindService()方法前服务已经被绑定，多次调用 bindService()方法并不会导致多次创建服务及绑定(也就是说 onCreate()和 onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务解除绑定，可以调用 unbindService()方法，调用该方法也会导致系统调用服务的 onUnbind()-->onDestroy()方法。

服务常用生命周期回调方法如下：

onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次 startService()或 bindService()方法，服务也只被创建一次。

onDestroy()该方法在服务被终止时调用。

与采用 Context.startService()方法启动服务有关的生命周期方法

onStart() 只有采用 Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。多次调用 startService()方法尽管不会多次创建服务，但 onStart() 方法会被多次调用。

与采用 Context.bindService()方法启动服务有关的生命周期方法

onBind()只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，当调用者与服务已经绑定，多次调用 Context.bindService()方法并不会导致该方法 onBind()被多次调用。

onUnbind()只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用

```
采用 Context. bindService()方法启动服务的代码如下：
public class HelloActivity extends Activity {
        ServiceConnection conn = new ServiceConnection() {
                public void onServiceConnected(ComponentName name, IBinder service)
{
                }
                public void onServiceDisconnected(ComponentName name) {
                }
        };
      @Override
public void onCreate(Bundle savedInstanceState) {
            Button button =(Button) this.findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener(){
                    public void onClick(View v) {
                        Intent intent = new Intent(HelloActivity.this, SMSService.class);
                        bindService(intent, conn, Context.BIND_AUTO_CREATE);
                        //unbindService(conn);//解除绑定
          }});
    }
}
```

19.ListView 的优化方案

1，如果自定义适配器，那么在 getView 方法中要考虑方法传进来的参数 contentView 是否为 null，如果为 null 就创建 contentView 并返回，如果不为 null 则直接使用。在这个方法中，尽可能少创建 view。

2，给 contentView 设置 tag(setTag()),传入一个 viewHolder 对象，用于缓存要显示的数据，可以达到图像数据异步加载的效果

3，如果 listview 需要显示的 item 很多，就要考虑分页加载。比如一共要显示100条或者更多的时候，我们可以考虑先加载20条，等用户拉到列表底部的时候，再去加载接下来的20条。

20广播接收者生命周期

广播接收器只有一个回调方法：

void onReceive(Context curContext, Intent broadcastMsg)

当广播消息抵达接收器时，Android 调用它的 onReceive()方法并将包含消息的 Intent 对象传递给它。广播接收器仅在它执行这个方法时处于活跃状态。当 onReceive()返回后，它即为失活状态。

拥有一个活跃状态的广播接收器的进程被保护起来而不会被杀死。但仅拥有失活状态组件的进程则会在其它进程需要它所占有的内存的时候随时被杀掉。

这种方式引出了一个问题：如果响应一个广播信息需要很长的一段时间，我们一般会将其纳入一个衍生的线程中去完成，而不是在主线程内完成它，从而保证用户交互过程的流畅。如果 onReceive()衍生了一个线程并且返回，则包涵新线程在内的整个进程都被会判为失活状态（除非进程内的其它应用程序组件仍处于活跃状态），于是它就有可能被杀掉。这个问题的解决方法是令 onReceive()启动一个新服务，并用其完成任务，于是系统就会知道进程中仍然在处理着工作。

21.设计模式和 IoC(Inversion of Control 控制反转)

Android 框架魅力的源泉在于 IoC，在开发 Android 的过程中你会时刻感受到 IoC 带来的巨大方便，就拿 Activity 来说，下面的函数是框架调用自动调用的：

protected void onCreate(Bundle savedInstanceState) ；

不是程序编写者主动去调用，反而是用户写的代码被框架调用，这也就反转了！当然 IoC 本身的内涵远远不止这些，但是从这个例子中也可以窥视出 IoC

带来的巨大好处。此类的例子在 Android 随处可见，例如说数据库的管理类，

例如说 Android 中 SAX 的 Handler 的调用等。有时候，您甚至需要自己编写简

单的 IoC 实现，上面展示的多线程现在就是一个说明

22.Android 中的长度单位详解

现在这里介绍一下 dp 和 sp。dp 也就是 dip。这个和 sp 基本类似。如果设置表示长度、

高度等属性时可以使用 dp 或 sp。但如果设置字体，需要使用 sp。dp 是与密度无关，sp

除了与密度无关外，还与 scale 无关。如果屏幕密度为160，这时 dp 和 sp 和 px 是一样

的。1dp=1sp=1px，但如果使用 px 作单位，如果屏幕大小不变（假设还是3.2 寸），而屏

幕密度变成了320。那么原来 TextView 的宽度设成160px，在密度为320 的3.2 寸屏幕里

看要比在密度为160 的3.2 寸屏幕上看短了一半。但如果设置成160dp 或160sp 的话。

系统会自动将 width 属性值设置成320px 的。也就是160 * 320 / 160。其中320 / 160 可

称为密度比例因子。

也就是说，如果使用 dp 和 sp，系统会根据屏幕密度的变化自动进行转换。

下面看一下其他单位的含义

px：表示屏幕实际的象素。例如，320*480 的屏幕在横向有320个象素，

在纵向有480 个象素。

in：表示英寸，是屏幕的物理尺寸。每英寸等于2.54 厘米。例如，形容

手机屏幕大小，经常说，3.2（英）寸、3.5（英）寸、4（英）寸就是指这个

单位。这些尺寸是屏幕的对角线长度。如果手机的屏幕是3.2 英寸，表示手机

的屏幕（可视区域）对角线长度是3.2*2.54 = 8.128 厘米。读者可以去量

一量自己的手机屏幕，看和实际的尺寸是否一致。

23. 4种 activity 的启动模式

standard: 标准模式，一调用 startActivity()方法就会产生一个新的实例。

singleTop: 如果已经有一个实例位于 Activity 栈的顶部时，就不产生新的实例，而只是调用 Activity 中的 newInstance()方法。如果不位于栈顶，会产生一个新的实例。

singleTask: 会在一个新的 task 中产生这个实例，以后每次调用都会使用这个，不会去产生新的实例了。

singleInstance: 这个跟 singleTask 基本上是一样，只有一个区别：在这个模式下的 Activity 实例所处的 task 中，只能有这个 activity 实例，不能有其他的实例。

24.什么是 ANR 如何避免它?

ANR：Application Not Responding。

在 Android 中，活动管理器和窗口管理器这两个系统服务负责监视应用程序的响应。当出现下列情况时，Android 就会显示 ANR 对话框了：

用户对应用程序的操作(如输入事件，按键、触摸屏事件)在5秒内无响应

广播接受器(BroadcastReceiver)在10秒内仍未执行完毕

Android 应用程序完全运行在一个独立的线程中(例如 main)。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发 ANR。因为此时，你的应用程序已经没有机会去响应输入事件和意向广播(Intent broadcast)。

避免方法：Activity 应该在它的关键生命周期方法（如 onCreate()和 onResume()）里尽可能少的去做创建操作，
潜在的耗时操作。例如网络或数据库操作，或者高耗时的计算如改变位图尺寸，应该在子线程里（或者异步方式）来完成。
主线程应该为子线程提供一个 Handler，以便完成时能够提交给主线程。

25.Android Intent 的使用

在一个 Android 应用中，主要是由一些组件组成，（Activity,Service,ContentProvider,etc.）在这些组件之间的通讯中，由 Intent 协助完成。

正如网上一些人解析所说，Intent 负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。Intent 在这里起着实现调用者与被调用者之间的解耦作用。
Intent 传递过程中，要找到目标消费者（另一个 Activity,IntentReceiver 或 Service），也就是 Intent 的响应者，有两种方法来匹配：
1，显示匹配（Explicit)：

```
20  public TestB extents Activity
21  {
22    .....
23  };
24    public class Test extends Activity
25  {
26          .....
27          public void switchActivity()
28          {
29                  Intent i = new Intent(Test.this, TestB.class);
30                  this.startActivity(i);
31          }
32  }
```

代码简洁明了，执行了 switchActivity()函数，就会马上跳转到名为 TestB 的 Activity 中。

2，隐式匹配(Implicit):

隐式匹配，首先要匹配 Intent 的几项值：Action, Category, Data/Type,Component 如果填写了 Componet 就是上例中的 Test.class)这就形成了显示匹配。所以此部分只讲前几种匹配。匹配规则为最大匹配规则，

1，如果你填写了 Action，如果有一个程序的 Manifest.xml 中的某一个 Activity 的 IntentFilter 段中定义了包含了相同的 Action 那么这个 Intent 就与这个目标 Action 匹配，如果这个 Filter 段中没有定义 Type,Category，那么这个 Activity 就匹配了。但是如果手机中有两个以上的程序匹配，那么就会弹出一个对话可框来提示说明。
Action 的值在 Android 中有很多预定义，如果你想直接转到你自己定义的 Intent 接收者，你可以在接收者的 IntentFilter 中加入一个自定义的 Action 值（同时要设定 Category 值为

"android.intent.category.DEFAULT")，在你的 Intent 中设定该值为 Intent 的 Action,就直接能跳转到你自己的 Intent 接收者中。因为这个 Action 在系统中是唯一的。

2,data/type，你可以用 Uri 来做为 data,比如 Uri uri = Uri.parse(http://www.google.com);Intent i = new Intent(Intent.ACTION_VIEW,uri);手机的 Intent 分发过程中，会根据 http://www.google.com 的 scheme 判断出数据类型 type

手机的 Brower 则能匹配它，在 Brower 的 Manifest.xml 中的 IntenFilter 中首先有 ACTION_VIEW Action,也能处理 http:的 type，

3，至于分类 Category，一般不要去在 Intent 中设置它，如果你写 Intent 的接收者，就在 Manifest.xml 的 Activity 的 IntentFilter 中包含 android.category.DEFAULT,这样所有不设置 Category（Intent.addCategory(String c);）的 Intent 都会与这个 Category 匹配。

4,extras（附加信息），是其它所有附加信息的集合。使用 extras 可以为组件提供扩展信息，比如，如果要执行"发送电子邮件"这个动作，可以将电子邮件的标题、正文等保存在 extras 里，传给电子邮件发送组件。

第四章 Android 的优点与不足

Android 平台手 机 5大优势：

一、开放性

在优势方面，Android 平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更 多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟

开发性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带 来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

二、挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年 iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约 减少。随着 EDGE、HSDPA 这些2G 至3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机 IM 软件方便地

进行 即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？

互联网巨头 Google 推动的 Android 终端天生就有网络特色，将让用户离互联网更近。

三、丰富的硬件选择

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异 和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian 风格手机 一下改用苹果 iPhone ， 同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移，是不是非常方便呢？

四、不受任何限制的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也 有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给 Android 难题之一。

五、无缝结合的 Google 应用

如今叱诧互联网的 Google 已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和 互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

再说 Android 的5大不足：

一、安全和隐私

由于手机 与互联网的紧密联 系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google 这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将会带来 新一轮的隐私危机。

二、首先开卖 Android 手机的不是最大运营商

众所周知，T-Mobile 在23日，于美国纽约发布 了 Android 首款手机 G1。但是在北美市场，最大的两家运营商乃 AT&T 和 Verizon，而 目前所知取得 Android 手机销售权的仅有 T-Mobile 和 Sprint，其中 T-Mobile 的3G 网络相对于其他三家也要逊色不少，因此，用户可以买账购买 G1，能否体验到最佳的3G 网络服务则要另当别论了！

三、运营商仍然能够影响到 Android 手机

在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。

这样的情况在国外市场同样出现。Android 手机的另一发售 运营商 Sprint 就将在其机型中内置其手机商店程序。

四、同类机型用户减少

在不少手机论坛 都会有针对某一型号 的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于 Android 平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少， 缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧 的机型以外。

五、过分依赖开发商缺少标准配置

在使用 PC 端的 Windows Xp 系统的时候，都会内置微软 Windows Media Player 这样一个浏览器程序，用户可以选择更多样的播放器，如 Realplay 或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android 平台中，由于其开放性，软件更多依赖第三方厂商，比如 Android 系统的 SDK 中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统 一性。

第五章 Android NDK

1、前言

6月 26 日， Google Android 发布了 NDK ，引起了很多发人员的兴趣。 NDK 全称：Native Development Kit。下载地址为：

http://developer.android.com/sdk/ndk/1.5_r1/index.html 。

2、误解

新出生的事物，除了惊喜外，也会给我们带来一定的迷惑、误解。

2.1、误解一： NDK 发布之前， Android 不

支持进行 C 开发

在 Google 中搜索“NDK”，很多“Android 终于可以使用 C++ 开发”之类的标题，这是一种对 Android 平台编程方式的误解。其实， Android 平台从诞生起，就已经支持 C 、C++ 开发。众所周知， Android 的 SDK 基于 Java 实现，这意味着基于 Android SDK 进行开发的第三方应用都必须使用 Java 语言。但这并不等同于“ 第三方应用只能使用 Java”。在 Android SDK 首次发布时， Google 就宣称其虚拟机 Dalvik 支持 JNI 编程方式，也就是第三方应用完全可以通过 JNI 调用自己的 C 动态库，即在 Android 平台上，“Java+C”的编程方式是一直都可以实现的。

当然这种误解的产生是有根源的：在 Android SDK 文档里，找不到任何 JNI 方面的帮助。即使第三方应用开发者使用 JNI 完成了自己的 C 动态链接库（ so ）开发，但是 so 如何和应用程序一起打包成 apk 并发布？这里面也存在技术障碍。我曾经花了不少时间，安装交叉编译器创建 so ，并通过 asset （资源）方式，实现捆绑 so 发布。但这种方式只能属于取巧的方式，并非官方支持。所以，在 NDK 出来之前，我们将“Java+C”的开发模式称之为灰色模式，即官方既不声明“ 支持这种方式 ”，也不声明“ 不支持这种方式 ”。

2.2、误解二：有了 NDK ，我们可以使用纯 C 开发 Android 应用

Android SDK 采用 Java 语言发布，把众多的 C 开发人员排除在第三方应用开发外（ 注意：我们所有讨论都是基于“ 第三方应用开发 ”， Android 系统基于 Linux ，系统级别的开发肯定是支持 C 语言的。 ）。NDK 的发布，许多人会误以为，类似于 Symbian 、WM ，在 Android 平台上终于可以使用纯 C 、C++ 开发第三方应用了！其实不然， NDK 文档明确说明： it is not a good way 。因为 NDK 并没有提供各种系统事件处理支持，也没有提供应用程序生命周期维护。此外，在本次发布的 NDK 中，应用程序 UI 方面的 API 也没有提供。至少目前来说，使用纯 C 、 C++ 开发一个完整应用的条件还不完备。

3、NDK 是什么

对 NDK 进行了粗略的研究后，我对“NDK 是什么 ”的理解如下：

1、NDK 是一系列工具的集合。

- NDK 提供了一系列的工具，帮助开发者快速开发 C （或 C++ ）的动态库，并能自动将 so 和 java 应用一起打包成 apk 。这些工具对开发者的帮助是巨大的。

- NDK 集成了交叉编译器，并提供了相应的 mk 文件隔离 CPU 、平台、 ABI 等差异，开发人员只需要简单修改 mk 文件（指出“ 哪些文件需要编译 ”、“ 编译特性要求 ”等），就可以创建出 so 。
- NDK 可以自动地将 so 和 Java 应用一起打包，极大地减轻了开发人员的打包工作。

2、NDK 提供了一份稳定、功能有限的 API 头文件声明。

Google 明确声明该 API 是稳定的，在后续所有版本中都稳定支持当前发布的 API 。从该版本的 NDK 中看出，这些 API 支持的功能非常有限，包含有： C 标准库（ libc ）、标准数学库（ libm ）、压缩库（ libz ）、 Log 库（ liblog ）。

4、NDK 带来什么

1、NDK 的发布，使“Java+C”的开发方式终于转正，成为官方支持的开发方式。

- 使用 NDK ，我们可以将要求高性能的应用逻辑使用 C 开发，从而提高应用程序的执行效率。
- 使用 NDK ，我们可以将需要保密的应用逻辑使用 C 开发。毕竟， Java 包都是可以反编译的。
- NDK 促使专业 so 组件商的出现。（乐观猜想，要视乎 Android 用户的数量）

2、NDK 将是 Android 平台支持 C 开发的开端。

NDK 提供了的开发工具集合，使开发人员可以便捷地开发、发布 C 组件。同时， Google 承诺在 NDK 后续版本中提高“ 可调式 ”能力，即提供远程的 gdb 工具，使我们可以便捷地调试 C 源码。在支持 Android 平台 C 开发，我们能感觉到 Google 花费了很大精力，我们有理由憧憬“C 组件支持 ”只是 Google Android 平台上 C 开发的开端。毕竟， C 程序员仍然是码农阵营中的绝对主力，将这部分人排除在 Android 应用开发之外，显然是不利于 Android 平台繁荣昌盛的。

第六章 Native 关键字的认识!

NI 是 J ava Native Interface 的 缩写。从 Java 1.1开始，Java Native Interface (JNI) 标准成为 java 平台的一部分，它允许 Java 代码和其他语言写的代码进行交互。JNI 一开始是为了本地已编译语言，尤其是 C 和 C++而设计 的，但是它并不妨碍你使用其他语言，

只要调用约定受支持就可以了。

使用 java 与本地已编译的代码交互，通常会丧失平台可移植性。但是，有些情况下这样做是可以接受的，甚至是必须的，比如，使用一些旧的库，与硬件、操作系统进行交互，或者为了提高程序 的性能。JNI 标准至少保证本地代码能工作在任何 Java 虚拟机实现下。

JNI（Java Native Interface）的书写步骤

·编写带有 native 声明的方法的 java 类

·使用 javac 命令编译所编写的 java 类

·使用 javah ?jni java 类名生成扩展名为 h 的头文件

·使用 C/C++（或者其他编程想语言）实现本地方法

·将 C/C++编写的文件生成动态连接库

1) 编写 java 程序：

这里以 HelloWorld 为例。

代码1：

```
class HelloWorld {
public native void displayHelloWorld();
static {
System.loadLibrary("hello");
}
public static void main(String[] args) {
new HelloWorld().displayHelloWorld();
}
}
```

声明 native 方法：如果你想将一个方法做为一个本地方法的话，那么你就必须声明改方法为 native 的，并且不能实现。其中方法的参数和返回值在后面讲述。

Load 动态库：System.loadLibrary("hello");加载动态库（我们可以这样理解：我们的方法 displayHelloWorld()没 有实现，但是我们在下面就直接使用了，所以必须在使用之前对它进行初始化）这里一般是以 static 块进行加载的。同时需要注意的是 System.loadLibrary();的参数“hello”是动态库的名字。

main()方法

2) 编译没有什么好说的了

```
javac HelloWorld.java
```

3) 生成扩展名为 h 的头文件

```
javah ?jni HelloWorld
```

头文件的内容：

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include
/* Header for class HelloWorld */
#ifndef _Included_HelloWorld
#define _Included_HelloWorld
#ifdef __cplusplus
extern "C" {
#endif
/*
* Class: HelloWorld
* Method: displayHelloWorld
* Signature: ()V
*/
JNIEXPORT void JNICALL Java_HelloWorld_displayHelloWorld
(JNIEnv *, jobject);
#ifdef __cplusplus
}
#endif
#endif
```

（这 里我们可以这样理解：这个 h 文件相当于我们在 java 里面的接口，这里声明了一个 Java_HelloWorld_displayHelloWorld (JNIEnv *, jobject);方法，然后在我们的本地方法里面实现这个方法，也就是说我们在编写 C/C++程序的时候所使用的方法名必须和这里的一致）。

4) 编写本地方法

实现和由 javah 命令生成的头文件里面声明的方法名相同的方法。

代码2：

```
1 #include
2 #include "HelloWorld.h"
3 #include
4 JNIEXPORT void JNICALL Java_HelloWorld_displayHelloWorld(JNIEnv *env,
jobject obj)
{
printf("Hello world!/n");
return;
}
```

注 意代码2中的第1行，需要将 jni.h（该文件可以在%JAVA_HOME%/include 文件夹下面找到）文件引入，因为在程序中的 JNIEnv、 jobject 等类型都 是在该头文件中定义的；另外在第2行需要将 HelloWorld.h 头文件引入（我是这么理解的：相当于我们在编写 java 程序的时候，实现一个接口的话 需要声明才可以，这里就是将 HelloWorld.h 头文件里面声明的方法加以实现。当然不一定是这样）。然后保存为 HelloWorldImpl.c 就 ok 了。

5) 生成动态库

这里以在 Windows 中为例，需要生成 dll 文件。在保存 HelloWorldImpl.c 文件夹下面，使用 VC 的编译器 cl 成。

```
cl -I%java_home%/include -I%java_home%/include/win32 -LD
HelloWorldImp.c -Fehello.dll
```

注 意：生成的 dll 文件名在选项-Fe 后面配置，这里是 hello，因为在 HelloWorld.java 文件中我们 loadLibary 的时候使用的名字 是 hello。当然这里修改之后那里也需要修改。另外需要将-I%java_home%/include -I%java_home%/include/win32参数加上，因为在第四步里面编写本地方法的时候引入了 jni.h 文件。

6) 运行程序

java HelloWorld 就 ok。

JNI（Java Native Interface）调用中考虑的问题

在首次使用 JNI 的时候有些疑问，后来在使用中一一解决，下面就是这些问题的备忘：

1。 java 和 c 是如何互通的？

其实不能互通的原因主要是数据类型的问题，jni 解决了这个问题，例如那个 c 文件中的 jstring 数据类型就是 java 传入的 String 对象 ，经过 jni 函数的转化就能成为 c 的 char*。

对应数据类型关系如下表：

Java 类型 本地 c 类型 说明

boolean jboolean 无符号，8 位

byte jbyte 无符号，8 位

char jchar 无符号，16 位

short jshort 有符号，16 位

int jint 有符号，32 位

long jlong 有符号，64 位

float jfloat 32 位

double jdouble 64 位

void void N/A

2. 如何将 java 传入的 String 参数转换为 c 的 char*，然后使用？

java 传入的 String 参数，在 c 文件中被 jni 转换为 jstring 的数据类型，在 c 文件中声明 char* test，然后 test = (char*)(*env)->GetStringUTFChars(env, jstring, NULL);

注意：test 使用完后，通知虚拟机平台相关代码无需再访问：

(*env)->ReleaseStringUTFChars(env, jstring, test);

3. 将 c 中获取的一个 char*的 buffer 传递给 java？

这个 char*如果是一般的字符串的话，作为 string 传回去就可以了。如果是含有'/0'的 buffer，最好作为 bytearray 传出，因为可以制定 copy 的 length，如果 copy 到 string，可能到'/0'就截断了。

有两种方式传递得到的数据：

一种是在 jni 中直接 new 一个 byte 数组，然后调用函数 (*env)->SetByteArrayRegion(env, bytearray, 0, len, buffer);将 buffer 的值 copy 到 bytearray 中，函数直接 return bytearray 就可以了。

一种是 return 错误号，数据作为参数传出，但是 java 的基本数据类型是传值，对象是传递的引用，所以将这个需要传出的 byte 数组用某个类包一下，如下：

```
class RetObj
{
public byte[] bytearray;
}
```

这个对象作为函数的参数 retobj 传出，通过如下函数将 retobj 中的 byte 数组赋值便于传出。代码如下：

```
jclass cls;
jfieldID fid;
jbyteArray bytearray;
bytearray = (*env)->NewByteArray(env,len);
(*env)->SetByteArrayRegion(env, bytearray, 0, len, buffer);
cls = (*env)->GetObjectClass(env, retobj);
fid = (*env)->GetFieldID(env, cls, "retbytes", "[B"]);
(*env)->SetObjectField(env, retobj, fid, bytearray);
```

4. 不知道占用多少空间的 buffer，如何传递出去呢？

在 jni 的 c 文件中 new 出空间，传递出去。java 的数据不初始化，指向传递出去的空间即可。

第七章 Android 面试题加强版（三）

26.如果后台的 Activity 由于某原因被系统回收了，如何在被系统回收之前保存当前状态？

当你的程序中某一个 Activity A 在运行时中，主动或被动地运行另一个新的 Activity B

这个时候 A 会执行

Java 代码

```
public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putLong("id", 1234567890);
}
```

B 完成以后又会来找 A, 这个时候就有两种情况，一种是 A 被回收，一种是没有被回收，被回

收的 A 就要重新调用 onCreate()方法，不同于直接启动的是这回 onCreate()里是带上参数 savedInstanceState，没被收回的就还是 onResume 就好了。

savedInstanceState 是一个 Bundle 对象，你基本上可以把他理解为系统帮你维护的一个 Map 对象。在 onCreate()里你可能会 用到它，如果正常启动 onCreate 就不会有它，所以用的时候要判断一下是否为空。

Java 代码

```
if(savedInstanceState != null){
        long id = savedInstanceState.getLong("id");
}
```

就像官方的 Notepad 教程 里的情况，你正在编辑某一个 note，突然被中断，那么就把这个 note 的 id 记住，再起来的时候就可以根据这个 id 去把那个 note 取出来，程序就完整 一些。

这也是看你的应用需不需要保存什么，比如你的界面就是读取一个列表，那就不需要特殊记住什么，哦， 没准你需要记住滚动条的位置...

27.如何退出 Activity

对于单一 Activity 的应用来说，退出很简单，直接 finish()即可。当然，也可以用 killProcess()和 System.exit()这样的方法。现提供几个方法，供参考：

1、抛异常强制退出：该方法通过抛异常，使程序 Force Close。验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出 Force Close 的窗口。

2、记录打开的 Activity：每打开一个 Activity，就记录下来。在需要退出时，关闭每一个 Activity 即可。

3、发送特定广播：在需要结束应用时，发送一个特定的广播，每个 Activity 收到广播后，关闭即可。

4、递归退出在打开新的 Activity 时使用 startActivityForResult，然后自己加标志，在 onActivityResult 中处理，递归关闭。除了第一个，都是想办法把每一个 Activity 都结束掉，间接达到目的。但是这样做同样不完美。你会发现，如果自己的应用程序对每一个 Activity 都设置了 nosensor，在两个 Activity 结束的间隙，sensor 可能有效了。但至少，我们的目的达到了，而且没有影响用户使用。为了编程方便，最好定义一个 Activity 基类，处理这些共通问题。

28.请解释下在单线程模型中 Message、Handler、Message Queue、Looper 之间的关系。

答：简单的说，Handler 获取当前线程中的 looper 对象，looper 用来从存放 Message 的 MessageQueue 中取出 Message，再有 Handler 进行 Message 的分发和处理.

Message Queue(消息队列)：用来存放通过 Handler 发布的消息，通常附属于某一个创建它的线程，可以通过 Looper.myQueue()得到当前线程的消息队列

Handler：可以发布或者处理一个消息或者操作一个 Runnable，通过 Handler 发布消息，消息将只会发送到与它关联的消息队列，然也只能处理该消息队列中的消息

Looper：是 Handler 和消息队列之间通讯桥梁，程序组件首先通过 Handler 把消息传递给 Looper，Looper 把消息放入队列。Looper 也把消息队列里的消息广播给所有的 Handler：Handler 接受到消息后调用 handleMessage 进行处理

Message：消息的类型，在 Handler 类中的 handleMessage 方法中得到单个的消息进行处理

在单线程模型下，为了线程通信问题，Android 设计了一个 Message Queue(消息队列)，线程间可以通过该 Message Queue 并结合 Handler 和 Looper 组件进行信息交换。下面将对它们进行分别介绍：

1. Message

Message 消息，理解为线程间交流的信息，处理数据后台线程需要更新 UI，则发送 Message 内含一些数据给 UI 线程。

2. Handler

Handler 处理者，是 Message 的主要处理者，负责 Message 的发送，Message 内容的执行处理。后台线程就是通过传进来的 Handler 对象引用来 sendMessage(Message)。而使用 Handler，需要 implement 该类的 handleMessage(Message)方法，它是处理这些 Message 的操作内容，例如 Update UI。通常需要子类化 Handler 来实现 handleMessage 方法。

3. Message Queue

Message Queue 消息队列，用来存放通过 Handler 发布的消息，按照先进先出执行。

每个 message queue 都会有一个对应的 Handler。Handler 会向 messagequeue 通过两种方法发送消息：sendMessage 或 post。这两种消息都会插在 message queue 队尾并按先进先出执行。但通过这两种方法发送的消息执行的方式略有不同：通过 sendMessage 发送的是一个 message 对象,会被 Handler 的 handleMessage()函数处理；而通过 post 方法发送的是一个 runnable 对象，则会自己执行。

4. Looper

Looper 是每条线程里的 Message Queue 的管家。Android 没有 Global 的 MessageQueue，而 Android 会自动替主线程(UI 线程)建立 Message Queue，但在子线程里并没有建立 Message Queue。所以调用 Looper.getMainLooper()得到的主线程的 Looper 不为 NULL，但调用 Looper.myLooper()得到当前线程的 Looper 就有可能为 NULL。对于子线程使用 Looper，API Doc 提供了正确的使用方法：这个 Message 机制的大概流程：

1. 在 Looper.loop()方法运行开始后，循环地按照接收顺序取出 Message Queue 里面的非 NULL 的 Message。

2. 一开始 Message Queue 里面的 Message 都是 NULL 的。当 Handler.sendMessage(Message)到 Message Queue，该函数里面设置了那个 Message 对象的 target 属性是当前的 Handler 对象。随后 Looper 取出了那个 Message，则调用 该 Message 的 target 指向的 Hander 的 dispatchMessage 函数对 Message 进行处理。在 dispatchMessage 方法里，如何处理 Message 则由用户指定，三个判断，优先级从高到低：

1) Message 里面的 Callback，一个实现了 Runnable 接口的对象，其中 run 函数做处

理工作；

2) Handler 里面的 mCallback 指向的一个实现了 Callback 接口的对象，由其 handleMessage 进行处理；

3) 处理消息 Handler 对象对应的类继承并实现了其中 handleMessage 函数，通过这个实现的 handleMessage 函数处理消息。

由此可见，我们实现的 handleMessage 方法是优先级最低的！

3. Handler 处理完该 Message (updateUI) 后，Looper 则设置该 Message 为 NULL，以便回收！

在网上有很多文章讲述主线程和其他子线程如何交互，传送信息，最终谁来执行处理信息之类的，个人理解是最简单的方法——判断 Handler 对象里面的 Looper 对象是属于哪条线程的，则由该线程来执行！

1. 当 Handler 对象的构造函数的参数为空，则为当前所在线程的 Looper；

2. Looper.getMainLooper()得到的是主线程的 Looper 对象，Looper.myLooper()得到的是当前线程的 Looper 对象。

29.你如何评价 Android 系统？优缺点。

答：Android 平台手机 5大优势：

一、开放性

在优势方面，Android 平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

二、挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年 iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些2G 至3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机 IM 软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头 Google 推动的 Android 终端天生就有网络特色，将让用户离互联网更近。

三、丰富的硬件选择

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件

的兼容，好比你从诺基亚 Symbian 风格手机 一下改用苹果 iPhone ，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移，是不是非常方便呢？

四、不受任何限制的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给 Android 难题之一。

五、无缝结合的 Google 应用

如今叱诧互联网的 Google 已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

再说 Android 的5大不足：

一、安全和隐私

由于手机 与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google 这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将会带来新一轮的隐私危机。

二、首先开卖 Android 手机的不是最大运营商

众所周知，T-Mobile 在23日，于美国纽约发布 了 Android 首款手机 G1。但是在北美市场，最大的两家运营商乃 AT&T 和 Verizon，而目前所知取得 Android 手机销售权的仅有 T-Mobile 和 Sprint，其中 T-Mobile 的3G 网络相对于其他三家也要逊色不少，因此，用户可以买账购买 G1，能否体验到最佳的3G 网络服务则要另当别论了！

三、运营商仍然能够影响到 Android 手机

在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android 手机的另一发售运营商 Sprint 就将在其机型中内置其手机商店程序。

四、同类机型用户减少

在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于 Android 平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。

五、过分依赖开发商缺少标准配置

在使用 PC 端的 Windows Xp 系统的时候，都会内置微软 Windows Media Player 这样一个播放器程序，用户可以选择更多样的播放器，如 Realplay 或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android 平台中，由于其开放性，软件更多依赖第三方厂商，比如 Android 系统的 SDK 中就没有内置音乐 播放器，全部依赖第三方开发，缺

少了产品的统一性。

30.谈谈 android 数据存储方式。

Android 提供了5种方式存储数据：

（1）使用 SharedPreferences 存储数据；它是 Android 提供的用来存储一些简单配置信息的一种机制，采用了 XML 格式将数据存储到设备中。只能在同一个包内使用，不能在不同的包之间使用。

（2）文件存储数据；文件存储方式是一种较常用的方法，在 Android 中读取/写入文件的方法，与 Java 中实现 I/O 的程序是完全一样的，提供了 openFileInput()和 openFileOutput()方法来读取设备上的文件。

（3）SQLite 数据库存储数据；SQLite 是 Android 所带的一个标准的数据库，它支持 SQL 语句，它是一个轻量级的嵌入式数据库。

（4）使用 ContentProvider 存储数据；主要用于应用程序之间进行数据交换，从而能够让其他的应用保存或读取此 Content Provider 的各种数据类型。

（5）网络存储数据；通过网络上提供给我们的存储空间来上传(存储)和下载(获取)我们存储在网络空间中的数据信息。

31. Android 中 Activity, Intent, Content Provider, Service 各有什么区别。

Activity： 活动，是最基本的 android 应用程序组件。一个活动就是一个用户可以操作的可视化用户界面，每一个活动都被实现为一个独立的类，并且从活动基类继承而来。

Intent： 意图，描述应用想干什么。最重要的部分是动作和动作对应的数据。

Content Provider：内容提供器，android 应用程序能够将它们的数据保存到文件、SQLite 数据库中，甚至是任何有效的设备中。当你想将你的应用数据和其他应用共享时，内容提供器就可以发挥作用了。

Service：服务，具有一段较长生命周期且没有用户界面的程序组件。

32.View, surfaceView, GLSurfaceView 有什么区别。

view 是最基础的，必须在 UI 主线程内更新画面，速度较慢。

SurfaceView 是 view 的子类，类似使用双缓机制，在新的线程中更新画面所以刷新界面速度比 view 快

GLSurfaceView 是 SurfaceView 的子类，opengl 专用的

33.Manifest.xml 文件中主要包括哪些信息？

manifest：根节点，描述了 package 中所有的内容。

uses-permission：请求你的 package 正常运作所需赋予的安全许可。

permission： 声明了安全许可来限制哪些程序能你 package 中的组件和功能。

instrumentation：声明了用来测试此 package 或其他 package 指令组件的代码。

application：包含 package 中 application 级别组件声明的根节点。

activity：Activity 是用来与用户交互的主要工具。

receiver：IntentReceiver 能使的 application 获得数据的改变或者发生的操作，即使它当前不在运行。

service：Service 是能在后台运行任意时间的组件。

provider：ContentProvider 是用来管理持久化数据并发布给其他应用程序使用的组件。

34.根据自己的理解描述下 Android 数字签名。

(1)所有的应用程序都必须有数字证书，Android 系统不会安装一个没有数字证书的应用程序

(2)Android 程序包使用的数字证书可以是自签名的，不需要一个权威的数字证书机构签名认证

(3)如果要正式发布一个 Android 程序，必须使用一个合适的私钥生成的数字证书来给程序签名，而不能使用 adt 插件或者 ant 工具生成的调试证书来发布。

(4)数字证书都是有有效期的，Android 只是在应用程序安装的时候才会检查证书的有效期。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。

35. AIDL 的全称是什么?如何工作?能处理哪些类型的数据？

AIDL 全称 Android Interface Definition Language（Android 接口描述语言）是一种借口描述语言; 编译器可以通过 aidl 文件生成一段代码，通过预先定义的接口达到两个进程内

部通信进程跨界对象访问的目的.AIDL 的 IPC 的机制和 COM 或 CORBA 类似，是基于接口的，但它是轻量级的。它使用代理类在客户端和实现层间传递值. 如果要使用 AIDL, 需要完成2件事情: 1. 引入 AIDL 的相关类.; 2. 调用 aidl 产生的 class.理论上，参数可以传递基本数据类型和 String, 还有就是 Bundle 的派生类，不过在 Eclipse 中,目前的 ADT 不支持 Bundle 做为参数,

具体实现步骤如下:

1、创建 AIDL 文件，在这个文件里面定义接口，该接口定义了可供客户端访问的方法和属性。

2、编译 AIDL 文件，用 Ant 的话，可能需要手动，使用 Eclipse plugin 的话,可以根据 adil 文件自动生产 java 文件并编译，不需要人为介入.

3、在 Java 文件中，实现 AIDL 中定义的接口. 编译器会根据 AIDL 接口，产生一个 JAVA 接口。这个接口有一个名为 Stub 的内部抽象类，它继承扩展了接口并实现了远程调用需要的几个方法。接下来就需要自己去实现自定义的几个接口了.

4、向客户端提供接口 ITaskBinder, 如果写的是 service，扩展该 Service 并重载 onBind () 方法来返回一个实现上述接口的类的实例。

5、在服务器端回调客户端的函数. 前提是当客户端获取的 IBinder 接口的时候,要去注册回调函数，只有这样，服务器端才知道该调用那些函数

AIDL 语法很简单,可以用来声明一个带一个或多个方法的接口，也可以传递参数和返回值。由于远程调用的需要，这些参数和返回值并不是任何类型.下面是些 AIDL 支持的数据类型:

1. 不需要 import 声明的简单 Java 编程语言类型(int,boolean 等)

2. String, CharSequence 不需要特殊声明

3. List, Map 和 Parcelables 类型，这些类型内所包含的数据成员也只能是简单数据类型，String 等其他比支持的类型.

(另外: 我没尝试 Parcelables, 在 Eclipse+ADT 下编译不过, 或许以后会有所支持).

实现接口时有几个原则:

.抛出的异常不要返回给调用者. 跨进程抛异常处理是不可取的.

.IPC 调用是同步的。如果你知道一个 IPC 服务需要超过几毫秒的时间才能完成地话，你应该避免在 Activity 的主线程中调用。也就是 IPC 调用会挂起应用程序导致界面失去响应. 这种情况应该考虑单起一个线程来处理.

.不能在 AIDL 接口中声明静态属性。

IPC 的调用步骤:

1. 声明一个接口类型的变量，该接口类型在.aidl 文件中定义。

2. 实现 ServiceConnection。

3. 调用 ApplicationContext.bindService(),并在 ServiceConnection 实现中进行传递.

4. 在 ServiceConnection.onServiceConnected()实现中，你会接收一个 IBinder 实例(被调用的 Service). 调用

YourInterfaceName.Stub.asInterface((IBinder)service)将参数转换为 YourInterface 类型。

5. 调用接口中定义的方法。你总要检测到 DeadObjectException 异常，该异常在连接断开时被抛出。它只会被远程方法抛出。

6. 断开连接，调用接口实例中的 ApplicationContext.unbindService()

参考：http://buaadallas.blog.51cto.com/399160/372090

36.android:gravity 与 android:layout_gravity 的区别

LinearLayout 有两个非常相似的属性：android:gravity 与 android:layout_gravity。他们的区别在 于：android:gravity 用于设置 View 组件的对齐方式，而 android:layout_gravity 用于设置 Container 组件的 对齐方式。

举个例子，我们可以通过设置 android:gravity="center"来让 EditText 中的文字在 EditText

组件中居中显示;同 时我们设置 EditText 的 android:layout_gravity="right"来让 EditText 组件在 LinearLayout 中居右 显示。来实践以下：

正如我们所看到的，在 EditText 中，其中的文字已经居中显示了，而 EditText 组件自己也对齐到了 LinearLayout 的右侧。

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <EditText
        android:layout_width="wrap_content"
        android:gravity="center"
        android:layout_height="wrap_content"
        android:text="one"
        android:layout_gravity="right"/>
</LinearLayout>
```

38. 注册广播接收者两种方式的区别，及优缺点

答：首先写一个类要继承 BroadcastReceiver

第一种:在清单文件中声明,添加

<receiveandroid:name=".IncomingSMSReceiver " >

<intent-filter>

<actionandroid:name="android.provider.Telephony.SMS_RECEIVED")

<intent-filter>

<receiver>

第二种使用代码进行注册如:

IntentFilterfilter = newIntentFilter("android.provider.Telephony.SMS_RECEIVED");

IncomingSMSReceiverreceiver = new IncomgSMSReceiver();

registerReceiver(receiver.filter);

两种注册类型的区别是：

1)第一种是常驻型（静态注册），也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

2)第二种不是常驻型广播（动态注册），也就是说广播跟随程序的生命周期。

注册的方法有两种，一种是静态注册，一种是动态注册。

动态注册优点：在 Android 的广播机制中，动态注册的优先级是要高于静态注册优先级的，因此在必要的情况下，我们是需要动态注册广播接收器的。

静态注册优点：动态注册广播接收器还有一个特点，就是当用来注册的 Activity 关掉后，广播也就失效了。同时反映了静态注册的一个优势，就是无需担忧广播接收器是否被关闭，只要设备是开启状态，广播接收器就是打开着的。

39.Dalvik 基于 JVM 的改进

1.几个 class 变为一个 dex，constant pool，省内存

2.Zygote，copy-on-write shared,省内存，省 cpu，省电

3.基于寄存器的 bytecode，省指令，省 cpu，省电

4.Trace-based JIT,省 cpu，省电,省内存

40.android 中有哪几种解析 xml 的类,官方推荐哪种？以及它们的原理和区别.

Ø DOM 解析

优点:

1.XML 树在内存中完整存储,因此可以直接修改其数据和结构.

2.可以通过该解析器随时访问 XML 树中的任何一个节点.

3.DOM 解析器的 API 在使用上也相对比较简单.

缺点:如果 XML 文档体积比较大时,将文档读入内存是非常消耗系统资源的.

使用场景:DOM 是用与平台和语言无关的方式表示 XML 文档的官方 W3C 标准.DOM 是以层次结构组织的节点的集合.这个层次结构允许开发人员在树中寻找特定信息.分析该结构通常需要加载整个文档和构造层次结构,然后才能进行任何工作.DOM 是基于对象层次结构的.

Ø SAX 解析

优点:

SAX 对内存的要求比较低,因为它让开发人员自己来决定所要处理的标签.特别是当开发人

员只需要处理文档中所包含的部分数据时,SAX 这种扩展能力得到了更好的体现.

缺点:

用 SAX 方式进行 XML 解析时,需要顺序执行,所以很难访问到同一文档中的不同数据.此外,在基于该方式的解析编码过程也相对复杂.

使用场景:

对于含有数据量十分巨大,而又不用对文档的所有数据进行遍历或者分析的时候,使用该方法十分有效.该方法不用将整个文档读入内存,而只需读取到程序所需的文档标签处即可.

Ø Xmlpull 解析

android SDK 提供了 xmlpull api,xmlpull 和 sax 类似,是基于流（stream）操作文件,然后根据节点事件回调开发者编写的处理程序.因为是基于流的处理,因此 xmlpull 和 sax 都比较节约内存资源,不会象 dom 那样要把所有节点以对橡树的形式展现在内存中.xmlpull 比 sax 更简明,而且不需要扫描完整个流.

41.Android 系统中 GC 什么情况下会出现内存泄露呢？

出现情况:

1. 数据库的 cursor 没有关闭

2.构造 adapter 时,没有使用缓存 contentview

衍生 listview 的优化问题----减少创建 view 的对象,充分使用 contentview,可以使用一静态类来优化处理 getview 的过程

3.Bitmap 对象不使用时采用 recycle()释放内存

4.activity 中的对象的生命周期大于 activity

调试方法: DDMS==> HEAPSZIE==>dataobject==>[Total Size]

42.谈谈对 Android NDK 的理解

NDK 全称：Native Development Kit。 1、NDK 是一系列工具的集合。 * NDK 提供了一系列的工具，帮助开发者快速开发 C（或 C++）的动态库，并能自动将 so 和 java 应用一起打包成 apk。这些工具对开发者的帮助是巨大的。 * NDK 集成了交叉编译器，并提供了相应的 mk 文件隔离 CPU、平台、ABI 等差异，开发人员只需要简单修改 mk 文件（指出“哪些文件需要编译”、“编译特性要求”等），就可以创建出 so。 * NDK 可以自动地将 so 和 Java 应用一起打包，极大地减轻了开发人员的打包工作。 2、NDK 提供了一份稳定、功能有限的 API 头文件声明。 Google 明确声明该 API 是稳定的，在后续所有版本中都稳定支持当前发布的 API。从该版本的 NDK 中看出，这些 API 支持的功能非常有限，包含有：C 标准库（libc）、标准数学库（libm）、压缩库（libz）、Log 库（liblog）。

第八章 Android 面试题加强版（四）

1，双缓冲技术原理以及优缺点：

创建一幅后台图像，将每一帧画入图像，然后调用 drawImage()方法将整个后台图像一次画到屏幕上去。

优点：双缓冲技术的优点在于大部分绘制是离屏的。

将离屏图像一次绘至屏幕上，比直接在屏幕上绘制要有效得多。

双缓冲技术可以使动画平滑。

缺点：要分配一个后台图像的缓冲，如果图像相当大，这将占用很大一块内存。

2，AsyncTask 简介

在开发 Android 移动客户端的时候往往要使用多线程来进行操作，我们通常会将耗时的操作放在单独的线程执行，避免其占用主线程而给用户带来不好的用户体验。但是在子线程中无法去操作主线程（UI 线程），在子线程中操作 UI 线程会出现错误。因此 android 提供了一个类 Handler 来在子线程中来更新 UI 线程，用发消息的机制更新 UI 界面，呈现给用户。这样就解决了子线程更新 UI 的问题。但是费时的任务操作总会启动一些匿名的子线程，太多的子线程给系统带来巨大的负担，随之带来一些性能问题。因此 android 提供了一个工具类 AsyncTask，顾名思义异步执行任务。这个 AsyncTask 生来就是处理一些后台的比较耗时的任务，给用户带来良好用户体验的，从编程的语法上显得优雅了许多，不再需要子线程和 Handler 就可以完成异步操作并且刷新用户界面。

3，Socket 通信编程

客户端编程步骤：

1、 创建客户端套接字(指定服务器端 IP 地址与端口号)

2、 连接(Android 创建 Socket 时会自动连接)

3、 与服务器端进行通信

4、 关闭套接字

服务器端:

1.创建一个 ServerSocket，用于监听客户端 Socket 的连接请求

2.采用循环不断接受来自客户端的请求

3.每当接受到客户端 Socket 的请求，服务器端也对应产生一个 Socket