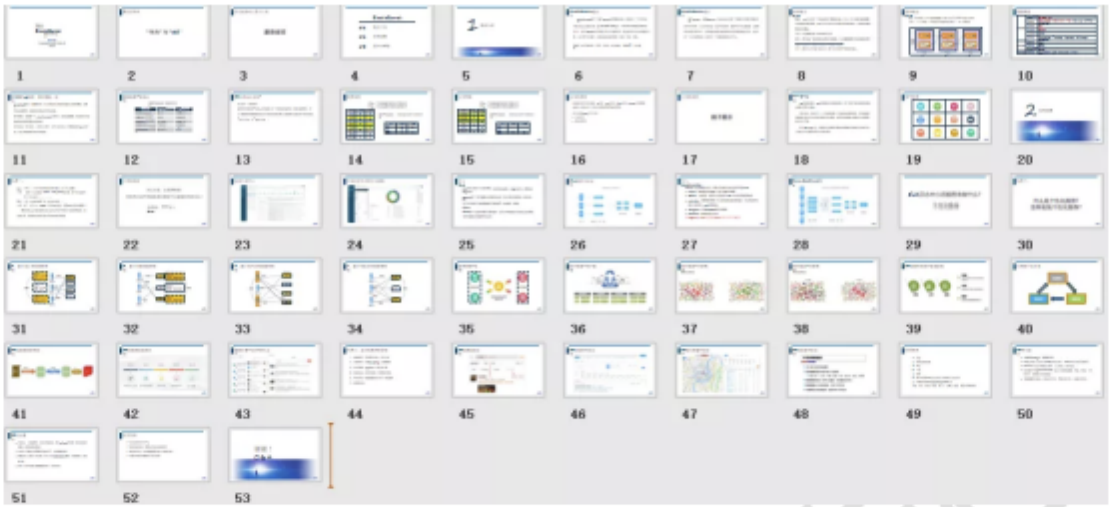
第二版：Android 40 道

目录

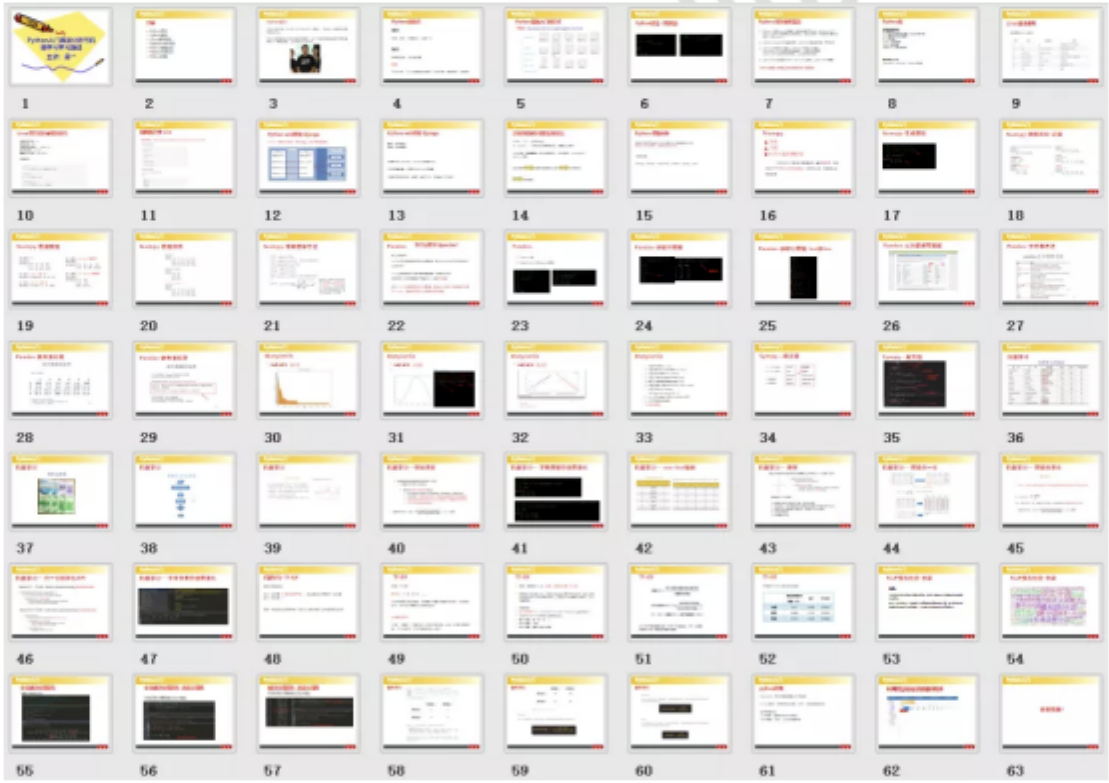
第二版：Android 40 道 1
1、 Android 的四大组件是哪些，它们的作用？ 3
2、 请介绍下 Android 中常用的五种布局。 3
3、 android 中的动画有哪几类，它们的特点和区别是什么 4
4、 android 中有哪几种解析 xml 的类？官方推荐哪种？以及它们的原理和区别。4
5、 ListView 的优化方案 5
6、 请介绍下 Android 的数据存储方式。 5
一：使用 SharedPreferences 存储数据 5
7、activity 的启动模式有哪些？是什么含义？ 10
8、跟 activity 和 Task 有关的 Intent 启动方式有哪些？其含义？ 12
9、 请描述下 Activity 的生命周期。 17
10、 activity 在屏幕旋转时的生命周期 17
11、 如何启用 Service，如何停用 Service。 17
12、 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。 19
第一种:在清单文件中声明 19
第二种使用代码进行注册如: 19
13、 请解释下在单线程模型中 Message、Handler、Message Queue、Looper 之间的关系。 20
14、 简要解释一下 activity、 intent 、intent filter、service、Broadcase、BroadcaseReceiver 23
15、 说说 mvc 模式的原理，它在 android 中的运用,android 的官方建议应用程序的开发采用 mvc 模式。何谓 mvc？ 24
16、 什么是 ANR 如何避免它？ 25
17、 什么情况会导致 Force Close ？如何避免？能否捕获导致其的异常？ 25
18、 描述一下 android 的系统架构 25
19、 请介绍下 ContentProvider 是如何实现数据共享的。 26
20、 Service 和 Thread 的区别？ 27

21、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？ 28
22、 IntentService 有何优点? 28
23、 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? 29
24、 如何将一个 Activity 设置成窗口的样式。 29
25、 如何退出 Activity? 如何安全退出已调用多个 Activity 的 Application? 29
1、抛异常强制退出： 30
2、记录打开的 Activity： 31
3、发送特定广播： 31
4、递归退出 31
26、 AIDL 的全称是什么？如何工作？能处理哪些类型的数据? 31
27、 请解释下 Android 程序运行时权限与文件系统权限的区别。 33
28、 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。 33
29、 android 系统的优势和不足 33
一、开放性 33
二、挣脱运营商的束缚 34
三、丰富的硬件选择 34
四、不受任何限制的开发商 34
五、无缝结合的 Google 应用 34
一、安全和隐私 35
二、首先开卖 Android 手机的不是最大运营商 35
三、运营商仍然能够影响到 Android 手机 35
四、同类机型用户减少 35
五、过分依赖开发商缺少标准配置 36
30、 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 36
31、 sim 卡的 EF 文件是什么？有何作用 36
32、 嵌入式操作系统内存管理有哪几种， 各有何特性 36
33、 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? 37
34、 一条最长的短信息约占多少 byte? 37
35、 如何将 SQLite 数据库(dictionary.db 文件)与 apk 文件一起发布 37

36、 如何将打开 res aw 目录中的数据库文件?.....37
37、 DDMS 和 TraceView 的区别?.....38
38、 java 中如何引用本地语言.....38
39、 谈谈 Android 的 IPC（进程间通信）机制.....38
40、 NDK 是什么.....39



63 页《Python 数据分析入门 PPT》



1、 Android 的四大组件是哪些，它们的作用?

答：Activity：Activity 是 Android 程序与用户交互的窗口，是 Android 构造块中最基本的一种，它需要为保持各界面的状态，做很多持久化的事情，妥善管理生命周期以及一些跳转逻辑

service：后台服务于 Activity，封装有一个完整的功能逻辑实现，接受上层指令，完成相关的事物，定义好需要接受的 Intent 提供同步和异步的接口

Content Provider：是 Android 提供的第三方应用数据的访问方案，可以派生 Content Provider 类，对外提供数据，可以像数据库一样进行选择排序，屏蔽内部数据的存储细节，向外提供统一的借口模型，大大简化上层应用，对数据的整合提供了更方便的途径

BroadCast Receiver：接受一种或者多种 Intent 作触发事件，接受相关消息，做一些简单处理，转换成一条 Notification，统一了 Android 的事件广播模型

2、 请介绍下 Android 中常用的五种布局。

常用五种布局方式，分别是：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）。

一、FrameLayout：所有东西依次都放在左上角，会重叠，这个布局比较简单，也只能放一点比较简单的东西。二、LinearLayout:线性布局,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。三、AbsoluteLayout：绝对布局用 X,Y 坐标来指定元素的位置，这种布局方式也比较简单，但是在屏幕旋转时，往往会出问题，而且多个元素的时候，计算比较麻烦。四、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、 android:layout_toLeftOf 相对于父元素的地方 android:layout_alignParentLeft、android:layout_alignParentRigh；五、TableLayout：表格布局，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。每一个布局都有自己适合的方式，这五个布局元素可以相互嵌套应用，做出美观的界面。

3、 android 中的动画有哪几类，它们的特点和区别是什么

答：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。

4、 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。

答：XML 解析主要有三种方式，SAX、DOM、PULL。常规在 PC 上开发我们使用 Dom 相对轻松些，但一些性能敏感的数据库或手机上还是主要采用 SAX 方式，SAX 读取是单向的，优点:不占内存空间、解析属性方便，但缺点就是对于套嵌多个分支来说处理不是很方便。而 DOM 方式会把整个 XML 文件加载到内存中去，这里 Android 开发网提醒大家该方法在查找方面可以和 XPath 很好的结合如果数据量不是很大推荐使用，而 PULL 常常用在 J2ME 对于节点处理比较好，类似 SAX 方式，同样很节省内存，在 J2ME 中我们经常使用的 KXML 库来解析。

5、 ListView 的优化方案

答：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。

6、 请介绍下 Android 的数据存储方式。

答：使用 SharedPreferences 存储数据；文件存储数据；SQLite 数据库存储数据；使用 ContentProvider 存储数据； 网络存储数据；

Preference，File， DataBase 这三种方式分别对应的目录是/data/data/Package Name/Shared_Pref, /data/data/Package Name/files, /data/data/Package Name/database 。

一：使用 SharedPreferences 存储数据

首先说明 SharedPreferences 存储方式，它是 Android 提供的用来存储一些简单配置信息的一种机制，例如：登录用户的用户名与密码。其采用了 Map 数据结构来存储数据，以键值的方式存储，可以简单的读取与写入，具体实例如下：

```
void ReadSharedPreferences() {
    String strName, strPassword;
    SharedPreferences user = getSharedPreferences( “user_info” ,
0);
    strName = user.getString( “NAME” , ” ” );
    strPassword = user getString( “PASSWORD” , ” ” );
}
void WriteSharedPreferences(String strName, String strPassword) {
    SharedPreferences user = getSharedPreferences( “user_info” ,
0);
    uer.edit();
    user.putString( “NAME” , strName);
    user.putString( “PASSWORD” , strPassword);
    user.commit();
}
```

数据读取与写入的方法都非常简单，只是在写入的时候有些区别：先调用 edit()使其处于编辑状态，然后才能修改数据，最后使用 commit()提交修改的数据。实际上 SharedPreferences 是采用了 XML 格式将数据存储到设备中，在 DDMS 中的 File Explorer 中的/data/data//shares_prefs 下。使用 SharedPreferences 是有些限制的：只能在同一个包内使用，不能在不同的包之间使用。 </package name>

二：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。具体实例如下:

```
String fn =  “moandroid.log”；
FileInputStream fis = openFileInput(fn);
FileOutputStream fos = openFileOutput(fn,Context.MODE_PRIVATE);
```

三：网络存储数据

网络存储方式，需要与 Android 网络数据包打交道，关于 Android 网络数据包的详细说明，请阅读 Android SDK 引用了 Java SDK 的哪些 package？。

四：ContentProvider

1、ContentProvider 简介

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，如：采用文件方式对外共享数据，需要进行文件操作读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的好处是统一了数据访问方式。

2、Uri 类简介

Uri 代表了要操作的数据，Uri 主要包含了两部分信息：1.需要操作的 ContentProvider ，2.对 ContentProvider 中的什么数据进行操作，一个 Uri 由以下几部分组成:

1.scheme：ContentProvider（内容提供者）的 scheme 已经由 Android 所规定为：content://...

2.主机名（或 Authority）：用于唯一标识这个 ContentProvider，外部调用者可以根据这个标识来找到它。

3.路径（path）：可以用来表示我们要操作的数据，路径的构建应根据业务而定，如下：

要操作 contact 表中 id 为 10 的记录，可以构建这样的路径:/contact/10

要操作 contact 表中 id 为 10 的记录的 name 字段， contact/10/name

要操作 contact 表中的所有记录，可以构建这样的路径:/contact?

要操作的数据不一定来自数据库，也可以是文件等他存储方式，如下：

要操作 xml 文件中 contact 节点下的 name 节点，可以构建这样的路径：/contact/name

如果要把一个字符串转换成 Uri，可以使用 Uri 类中的 parse()方法，如下：

```
Uri uri =
Uri.parse("content://com.changcheng.provider.contactprovider/contact")
```

3、UriMatcher、ContentUrist 和 ContentResolver 简介

因为 Uri 代表了要操作的数据，所以我们很经常需要解析 Uri，并从 Uri 中获取数据。Android 系统提供了两个用于操作 Uri 的工具类，分别为 UriMatcher 和 ContentUris 。掌握它们的使用，会便于我们的开发工作。

UriMatcher：用于匹配 Uri，它的用法如下：

1.首先把你需要匹配 Uri 路径全部给注册上，如下：

```
//常量 UriMatcher.NO_MATCH 表示不匹配任何路径的返回码(-1)。
UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
//如果 match()方法匹配
content://com.changcheng.sqlite.provider.contactprovider /contact 路径，
返回匹配码为 1
uriMatcher.addURI(“com.changcheng.sqlite.provider.contactprovider”，
“contact”，1);//添加需要匹配 uri，如果匹配就会返回匹配码
//如果 match()方法匹配
content://com.changcheng.sqlite.provider.contactprovider/contact/230
路径，返回匹配码为 2
uriMatcher.addURI(“com.changcheng.sqlite.provider.contactprovider”，
“contact/#”，2);//#号为通配符
```

2.注册完需要匹配的 Uri 后，就可以使用 uriMatcher.match(uri)方法对输入的 Uri 进行匹配，如果匹配就返回匹配码，匹配码是调用 addURI()方法传入的第三个参数，假设匹配 content://com.changcheng.sqlite.provider.contactprovider/contact 路径，返回的匹配码为 1。

ContentUris：用于获取 Uri 路径后面的 ID 部分，它有两个比较实用的方法：

withAppendedId(uri, id)用于为路径加上 ID 部分

parseId(uri)方法用于从路径中获取 ID 部分

ContentResolver：当外部应用需要对 ContentProvider 中的数据进行添加、删除、修改和查询操作时，可以使用 ContentResolver 类来完成，要获取

ContentResolver 对象，可以使用 Activity 提供的 getContentResolver()方法。ContentResolver 使用 insert、delete、update、query 方法，来操作数据。

7、activity 的启动模式有哪些？是什么含义？

答：在 android 里，有 4 种 activity 的启动模式，分别为：

“standard” (默认)

“singleTop”

“singleTask”

“singleInstance”

它们主要有如下不同：

1. 如何决定所属 task

“standard”和”singleTop”的 activity 的目标 task，和收到的 Intent 的发送者在同一个 task 内，除非 intent 包括参数 FLAG_ACTIVITY_NEW_TASK。

如果提供了 FLAG_ACTIVITY_NEW_TASK 参数，会启动到别的 task 里。

“singleTask”和”singleInstance”总是把 activity 作为一个 task 的根元素，他们不会被启动到一个其他 task 里。

2. 是否允许多个实例

“standard”和”singleTop”可以被实例化多次，并且存在于不同的 task 中，且一个 task 可以包括一个 activity 的多个实例；

“singleTask”和”singleInstance”则限制只生成一个实例，并且是 task 的根元素。singleTop 要求如果创建 intent 的时候栈顶已经有要创建的 Activity 的实例，则将 intent 发送给该实例，而不发送给新的实例。

3. 是否允许其它 activity 存在于本 task 内

“singleInstance”独占一个 task，其它 activity 不能存在那个 task 里；如果它启动了一个新的 activity，不管新的 activity 的 launch mode 如何，新的 activity 都将会到别的 task 里运行（如同加了 FLAG_ACTIVITY_NEW_TASK 参数）。

而另外三种模式，则可以和其它 activity 共存。

4. 是否每次都生成新实例

“standard”对于没一个启动 Intent 都会生成一个 activity 的新实例；

“singleTop”的 activity 如果在 task 的栈顶的话，则不生成新的该 activity 的实例，直接使用栈顶的实例，否则，生成该 activity 的实例。

比如现在 task 栈元素为 A-B-C-D（D 在栈顶），这时候给 D 发一个启动 intent，如果 D 是 “standard”的，则生成 D 的一个新实例，栈变为 A - B - C - D - D。

如果 D 是 singleTop 的话，则不会生产 D 的新实例，栈状态仍为 A-B-C-D

如果这时候给 B 发 Intent 的话，不管 B 的 launchmode 是”standard” 还是“singleTop” ，都会生成 B 的新实例，栈状态变为 A-B-C-D-B。

“singleInstance”是其所在栈的唯一 activity，它会每次都被重用。

“singleTask”如果在栈顶，则接受 intent，否则，该 intent 会被丢弃，但是该 task 仍会回到前台。

当已经存在的 activity 实例处理新的 intent 时候，会调用 onNewIntent()方法 如果收到 intent 生成一个 activity 实例，那么用户可以通过 back 键回到上一个状态；如果是已经存在的一个 activity 来处理这个 intent 的话，用户不能通过按 back 键返回到这之前的状态。

8、跟 activity 和 Task 有关的 Intent 启动方式有哪些？其含义？

核心的 Intent Flag 有
FLAG_ACTIVITY_NEW_TAS
FLAG_ACTIVITY_CLEAR_TOP
FLAG_ACTIVITY_RESET_TASK_IF_NEEDED
FLAG_ACTIVITY_SINGLE_TO
FLAG_ACTIVITY_NEW_TAS

如果设置，这个 Activity 会成为历史 stack 中一个新 Task 的开始。一个 Task（从启动它的 Activity 到下一个 Task 中的 Activity）定义了用户可以迁移的 Activity 原子组。Task 可以移动到前台和后台；在某个特定 Task 中的所有 Activity 总是保持相同的次序

这个标志一般用于呈现“启动”类型的行为：它们提供用户一系列可以单独完成的事情，与启动它们的 Activity 完全无关。

使用这个标志，如果正在启动的 Activity 的 Task 已经在运行的话，那么，新的 Activity 将不会启动；代替的，当前 Task 会简单的移入前台。参考 FLAG_ACTIVITY_MULTIPLE_TASK 标志，可以禁用这一行为。

这个标志不能用于调用方对已经启动的 Activity 请求结果。

FLAG_ACTIVITY_CLEAR_TOP

如果设置，并且这个 Activity 已经在当前的 Task 中运行，因此，不再是重新启动一个这个 Activity 的实例，而是在这个 Activity 上方的所有 Activity 都将关闭，然后这个 Intent 会作为一个新的 Intent 投递到老的 Activity（现在位于顶端）中。

例如，假设一个 Task 中包含这些 Activity：A，B，C，D。如果 D 调用了 startActivity()，并且包含一个指向 Activity B 的 Intent，那么，C 和 D 都将结束，然后 B 接收到这个 Intent，因此，目前 stack 的状况是：A，B。

上例中正在运行的 Activity B 既可以在 onNewIntent()中接收到这个新的 Intent，也可以把自己关闭然后重新启动来接收这个 Intent。如果它的启动模式声明为“multiple”（默认值），并且你没有在这个 Intent 中设置 FLAG_ACTIVITY_SINGLE_TOP 标志，那么它将关闭然后重新创建；对于其它的启动模式，或者在这个 Intent 中设置 FLAG_ACTIVITY_SINGLE_TOP 标志，都将把这个 Intent 投递到当前这个实例的 onNewIntent()中。

这个启动模式还可以与 FLAG_ACTIVITY_NEW_TASK 结合起来使用：用于启动一个 Task 中的根 Activity，它会把那个 Task 中任何运行的实例带入前台，然后清除它直到根 Activity。这非常有用，例如，当从 Notification Manager 处启动一个 Activity。

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED

如果设置这个标志，这个 activity 不管是从一个新的栈启动还是从已有栈推到栈顶，它都将以 the front door of the task 的方式启动。这就讲导致任何与应用相关的栈都讲重置到正常状态（不管是正在讲 activity 移入还是移除），如果需要，或者直接重置该栈为初始状态。

FLAG_ACTIVITY_SINGLE_TOP

如果设置，当这个 Activity 位于历史 stack 的顶端运行时，不再启动一个新的

FLAG_ACTIVITY_BROUGHT_TO_FRONT

这个标志一般不是由程序代码设置的，如在 launchMode 中设置 singleTask 模式时系统帮你设定。

FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET

如果设置，这将在 Task 的 Activity stack 中设置一个还原点，当 Task 恢复时，需要清理 Activity。也就是说，下一次 Task 带着 FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 标记进入前台时（典型的操作是用户在主画面重启它），这个 Activity 和它之上的都将关闭，以至于用户不能再返回到它们，但是可以回到之前的 Activity。

这在你的程序有分割点的时候很有用。例如，一个 e-mail 应用程序可能有一个操作是查看一个附件，需要启动图片浏览 Activity 来显示。这个 Activity 应该作为 e-mail 应用程序 Task 的一部分，因为这是用户在这个 Task 中触发的操作。然而，当用户离开这个 Task，然后从主画面选择 e-mail app，我们可能希望回到查看的会话中，但不是查看图片附件，因为这让人困惑。通过在启动图片浏览时设定这个标志，浏览及其它启动的 Activity 在下次用户返回到 mail 程序时都将全部清除。

FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS

如果设置，新的 Activity 不会在最近启动的 Activity 的列表中保存。

FLAG_ACTIVITY_FORWARD_RESULT

如果设置，并且这个 Intent 用于从一个存在的 Activity 启动一个新的 Activity，那么，这个作为答复目标的 Activity 将会传到这个新的 Activity 中。这种方式下，新的 Activity 可以调用 setResult(int)，并且这个结果值将发送给那个作为答复目标的 Activity。

FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY

这个标志一般不由应用程序代码设置，如果这个 Activity 是从历史记录里启动的（常按 HOME 键），那么，系统会帮你设定。

FLAG_ACTIVITY_MULTIPLE_TASK

不要使用这个标志，除非你自己实现了应用程序启动器。与 FLAG_ACTIVITY_NEW_TASK 结合起来使用，可以禁用把已存的 Task 送入前台的行为。当设置时，新的 Task 总是会启动来处理 Intent，而不管这是是否已经有一个 Task 可以处理相同的事情。

由于默认的系统不包含图形 Task 管理功能，因此，你不应该使用这个标志，除非你提供给用户一种方式可以返回到已经启动的 Task。

如果 FLAG_ACTIVITY_NEW_TASK 标志没有设置，这个标志被忽略。

FLAG_ACTIVITY_NO_ANIMATION

如果在 Intent 中设置，并传递给 Context.startActivity()的话，这个标志将阻止系统进入下一个 Activity 时应用 Acitivity 迁移动画。这并不意味着动画将永不运行——如果另一个 Activity 在启动显示之前，没有指定这个标志，那么，动画将

被应用。这个标志可以很好的用于执行一连串的操作，而动画被看作是更高一级的事件的驱动。

FLAG_ACTIVITY_NO_HISTORY

如果设置，新的 Activity 将不再历史 stack 中保留。用户一离开它，这个 Activity 就关闭了。这也可以通过设置 noHistory 特性。

FLAG_ACTIVITY_NO_USER_ACTION

如果设置，作为新启动的 Activity 进入前台时，这个标志将在 Activity 暂停之前阻止从最前方的 Activity 回调的 onUserLeaveHint()。

典型的，一个 Activity 可以依赖这个回调指明显式的用户动作引起的 Activity 移出后台。这个回调在 Activity 的生命周期中标记一个合适的点，并关闭一些 Notification。

如果一个 Activity 通过非用户驱动的事件，如来电或闹钟，启动的，这个标志也应该传递给 Context.startActivity，保证暂停的 Activity 不认为用户已经知晓其 Notification。

FLAG_ACTIVITY_PREVIOUS_IS_TOP

If set and this intent is being used to launch a new activity from an existing one, the current activity will not be counted as the top activity for deciding whether the new intent should be delivered to the top instead of starting a new one. The previous activity will be used as the top, with the assumption being that the current activity will finish itself immediately.

FLAG_ACTIVITY_REORDER_TO_FRONT

如果在 Intent 中设置，并传递给 Context.startActivity()，这个标志将引发已经运行的 Activity 移动到历史 stack 的顶端。

例如，假设一个 Task 由四个 Activity 组成：A,B,C,D。如果 D 调用 startActivity()来启动 Activity B，那么，B 会移动到历史 stack 的顶端，现在的次序变成 A,C,D,B。如果 FLAG_ACTIVITY_CLEAR_TOP 标志也设置的话，那么这个标志将被忽略。

9、 请描述下 Activity 的生命周期。

答:activity 的生命周期方法有:onCreate()、onStart()、onReStart()、onResume()、onPause()、onStop()、onDestory();

可见生命周期：从 onStart()直到系统调用 onStop()

前台生命周期：从 onResume()直到系统调用 onPause()

10、 activity 在屏幕旋转时的生命周期

答：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；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；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

11、 如何启用 Service，如何停用 Service。

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 文件中的节点里对服务进行配置:

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

12、 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。

答：首先写一个类要继承 BroadcastReceiver

第一种:在清单文件中声明

第二种使用代码进行注册如:

```
IntentFilter filter =  new
IntentFilter("android.provider.Telephony.SMS_RECEIVED");
IncomingSMSReceiver receiver = new IncomgSMSReceiver();
registerReceiver(receiver.filter);
```

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。

13、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 message queue 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 Message Queue，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 (update UI) 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。

14、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，而是在后台运行，可以与一个 activity 绑定，通过绑定暴露出来接口并与其进行通信；一个 broadcast receiver 是一个接收广播消息并做出回应的 component，broadcast receiver 没有界面；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。

15、 说说 mvc 模式的原理，它在 android 中的运用,android 的官方建议应用程序的开发采用 mvc 模式。何谓 mvc?

mvc 是 model,view,controller 的缩写， 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，如果你对 android 了解的比较的多了话，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 交割 model 业务逻辑层处理，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。

16、 什么是 ANR 如何避免它?

答：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，当用户操作的在 5s 内应用程序没能做出反应，BroadcastReceiver 在 10 秒内没有执行完毕，就会出现应用程序无响应对话框，这既是 ANR。

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。

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

答：程序出现异常，比如 nullpointer。

避免：编写程序时逻辑连贯，思维缜密。能捕获异常，在 logcat 中能看到异常信息

18、 描述一下 android 的系统架构

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。

19、 请介绍下 ContentProvider 是如何实现数据共享的。

一个程序可以通过实现一个 Content provider 的抽象接口将自己的数据完全暴露出去，而且 Content providers 是以类似数据库中表的方式将数据暴露。Content providers 存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。

要想使应用程序的数据公开化，可通过 2 种方法：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。

如何通过一套标准及统一的接口获取其他应用程序暴露的数据？

Android 提供了 ContentResolver，外界的程序可以通过 ContentResolver 接口访问 ContentProvider 提供的数据。

20、 Service 和 Thread 的区别？

答：servie 是系统的组件，它由系统进程托管（servicemanager）；它们之间的通信类似于 client 和 server，是一种轻量级的 ipc 通信，这种通信的载体是 binder，它是在 linux 层交换信息的一种 ipc。而 thread 是由本应用程序托管。1). Thread：Thread 是程序执行的最小单元，它是分配 CPU 的基本单位。可以用 Thread 来执行一些异步的操作。

2). Service：Service 是 android 的一种机制，当它运行的时候如果是 Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是 Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上。

既然这样，那么我们为什么要用 Service 呢？其实这跟 android 的系统机制有关，我们先拿 Thread 来说。Thread 的运行是独立于 Activity 的，也就是说当

一个 Activity 被 finish 之后，如果你没有主动停止 Thread 或者 Thread 里的 run 方法没有执行完毕的话，Thread 也会一直执行。因此这里会出现一个问题：当 Activity 被 finish 之后，你不再持有该 Thread 的引用。另一方面，你没有办法在不同的 Activity 中对同一 Thread 进行控制。

举个例子：如果你的 Thread 需要不停地隔一段时间就要连接服务器做某种同步的话，该 Thread 需要在 Activity 没有 start 的时候也在运行。这个时候当你 start 一个 Activity 就没有办法在该 Activity 里面控制之前创建的 Thread。因此你便需要创建并启动一个 Service ，在 Service 里面创建、运行并控制该 Thread，这样便解决了该问题（因为任何 Activity 都可以控制同一 Service，而系统也只会创建一个对应 Service 的实例）。

因此你可以把 Service 想象成一种消息服务，而你可以在任何有 Context 的地方调用 Context.startService、Context.stopService、Context.bindService，Context.unbindService，来控制它，你也可以在 Service 里注册 BroadcastReceiver，在其他地方通过发送 broadcast 来控制它，当然这些都是 Thread 做不到的。

21、 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？

答：会，比如 nullpointerException。我遇到过，比如 textview.setText()时，textview 没有初始化。会导致程序无法正常运行出现 forceclose。打开控制台查看 logcat 信息找出异常信息并修改程序。

22、 IntentService 有何优点？

答: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

23、 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?

答：重写 onSaveInstanceState()方法，在此方法中保存需要保存的数据，该方法将会在 activity 被回收之前调用。通过重写 onRestoreInstanceState()方法可以从中提取保存好的数据

24、 如何将一个 Activity 设置成窗口的样式。

答：中配置： android :theme="@android:style/Theme.Dialog"

另外 android:theme="@android:style/Theme.Translucent" 是设置透明

25、 如何退出 Activity? 如何安全退出已调用多个 Activity 的 Application?

答：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。

对于多个 activity，1、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。2、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。3、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。为了编程方便，最好定义一个 Activity 基类，处理这些共通问题。

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，killBackground Processes()，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。

26、 AIDL 的全称是什么？如何工作？能处理哪些类型的数据?

答：全称是：Android Interface Define Language

在 Android 中，每个应用程序都可以有自己的进程．在写 UI 应用的时候，经常要用到 Service．在不同的进程中，怎样传递对象呢?显然, Java 中不允许跨进程内存共享．因此传递对象，只能把对象拆分成操作系统能理解的简单形式，以达到跨界对象访问的目的．在 J2EE 中,采用 RMI 的方式，可以通过序列化传递对象．在 Android 中，则采用 AIDL 的方式．理论上 AIDL 可以传递 Bundle,实际上做起来却比较麻烦。

AIDL(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的目的．如果需要在一个 Activity 中，访问另一个 Service 中的某个对象，需要先将对象转化成 AIDL 可识别的参数(可能是多个参数)，然后使用 AIDL 来传递这些参数，在消息的接收端，使用这些参数组装成自己需要的对象．

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.

AIDL 的创建方法:

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)

27、 请解释下 Android 程序运行时权限与文件系统权限的区别。

答：运行时权限 Dalvik(android 授权)

文件系统 linux 内核授权

28、 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

29、 android 系统的优势和不足

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。

30、 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

答：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。

31、 sim 卡的 EF 文件是什么？有何作用

答：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

32、 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

33、 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。

34、 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。

35、 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

36、 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。

37、 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 。

38、 java 中如何引用本地语言

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

39、 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。

40、 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