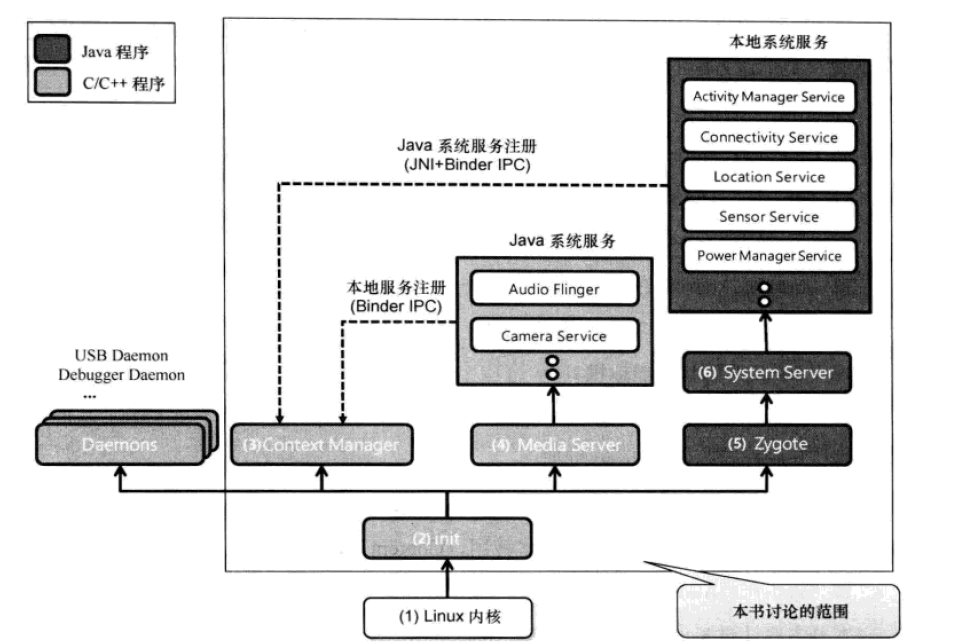
Android 笔记

源码下载手顺：

https://www.cnblogs.com/it-tsz/p/12175237.html

## 1 Android的启动过程



如图，

1. 在android启动时，通过bootloader进行先初始化linux内核，Android 属于嵌入式设备，并没有像 PC 那样的 BIOS 程序。 取而代之的是 Bootloader —— 系统启动加载器。使用bootloader系统启动加载器启动android内核。bootloader程序可以在\bootable\bootloader\legacy\usbloader找到。
2. Android内核启动后，调用init进程，init进程是android所有进程的父进程，是一个由内核启动的用户级的进程，是一个守护进程。init进程启动后，创建多个守护进程，（usb/debug/radio/…）对各种设备进行初始化。其中，init进程fork出Zygote进程，再有Zygote进程产生其他进程。

详细过程参考：<https://www.jianshu.com/p/75039ea24937>

## 2 阅读项目源码的大概思路

1、使用这个应用或者游戏，直到熟悉它的使用流程和功能

2、根据自己的体验分析代码实现（整体）

3、如果有条件和原作者请教项目思路

4、从最先启动的Activity开始，弄懂每个Activity功能如何实现以及它的布局实现

5、遇到不懂的代码，首先找搜索引擎（一般都能解决）。还不行就可以请教他人了

6、对新学到的知识点，一定要总结下来，并且定期的去回顾

7、最重要的一点：我认为要自己动手写，那怕是重新写一遍这个项目，或者自己设计一个类似的，再或者只写这个项目中的一部分。

## 3 Android源码结构





更详细目录结构介绍参考《android源码结构分析.txt》文件。

## 4 android源码和android内核源码的区别

Android源码编译之后生成的是ramdisk.img、system.img和userdata.img。而内核源码编译完成之后生成的是ZImage。在一般情况下Android源码是不带有内核源码的，但是带有一个镜像，这样在编译完Android源码之后就可以模拟器启动了，如果要更换系统的内核，此时将高版本的内核源码进行编译生成ZImage然后替换Android系统的的镜像。这样使用模拟器启动之后就可以查看内核是否已经被刷新。  
请注意，android源码和kernel源码是分开下载的

下载、编译android源码后，可以将其放在模拟器中执行，但是linux内核放在哪里？

下载好的源码是否包含了android内核？？

## 5 android中的每个系统服务是单独的一个进程吗？

系统服务都在SystemServer进程，每个服务都运行在自己的线程上

他们之间通过ServiceManager相互调用，serviceManager是一个独立进程，通过binder去调用。

## 6 阅读项目源码的大概思路

1、使用这个应用或者游戏，直到熟悉它的使用流程和功能

2、根据自己的体验分析代码实现（整体）

3、如果有条件和原作者请教项目思路

4、从最先启动的Activity开始，弄懂每个Activity功能如何实现以及它的布局实现

5、遇到不懂的代码，首先找搜索引擎（一般都能解决）。还不行就可以请教他人了

6、对新学到的知识点，一定要总结下来，并且定期的去回顾

7、最重要的一点：我认为要自己动手写，那怕是重新写一遍这个项目，或者自己设计一个类似的，再或者只写这个项目中的一部分。

## 7 设备、设备节点、设备驱动

## 8 android中的init

### 8.1 init的作用（四个）

1) 分析并运行init.rc文件

2) 生成设备驱动节点

3) 处理子进程的终止

4) 属性服务

### 8.2 init.rc 文件介绍

动作列表-on

服务列表-service

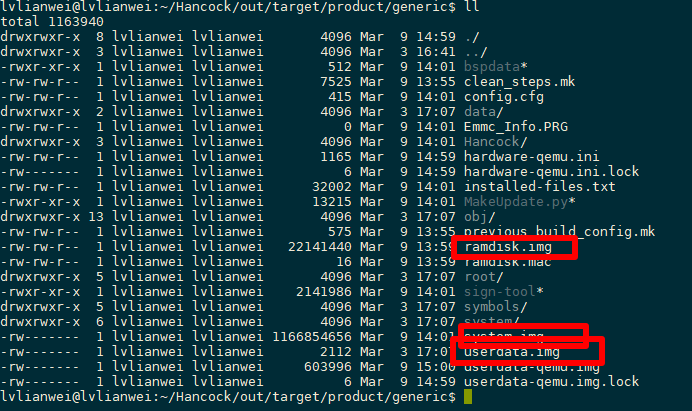
## 9 android源码编译产物

Android源码编译后有system.img,ramdisk.img,userdata.img三个映像文件，

Ramdisk.img ，ramdisk.img是文件系统，system.img包括了主要的包、库等文件，userdata.img包括了一些用户数据.

当emulator加载这3个映像文件后，会把system.img和userdata.img分别加载到ramdisk文件系统中的system和userdata目录下。

其中ramdisk.img是一个分区映像文件，即是一个文件系统（root file system），，kernel启动时，ramdisk负责init，system.img包括了主要系统，系统app等，挂载于文件系统下的system文件夹, userdata.img包括了一些用户数据，被挂载到文件系统下的data文件夹下。如下图所示：



## 10 android中的服务种类

Android中的服务一般分为两大类，

系统服务，adroid自带的服务，系统提供的。

应用程序服务，是开发者继承并实现service类后开发出的服务。（开发人员开发的）

系统服务——java系统服务：（个人理解）Android系统中用Java开发的系统服务

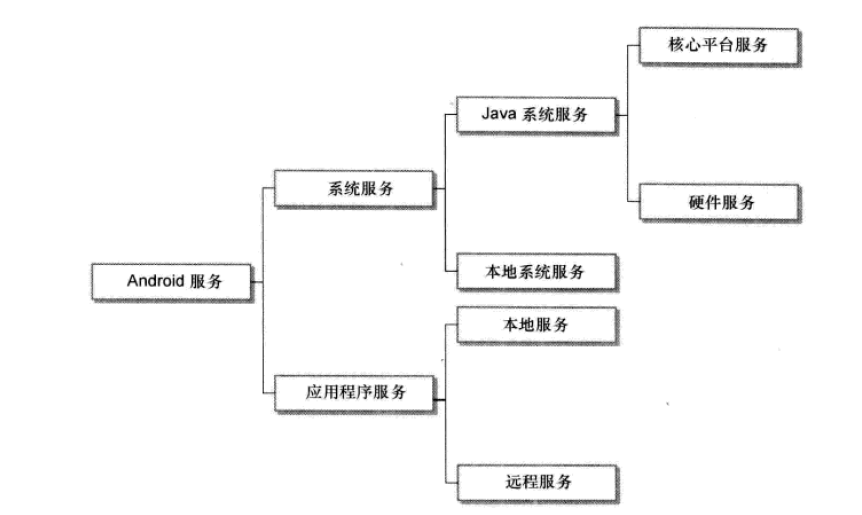
系统服务——本地系统服务：（个人理解）android系统中用C/C++开发的系统服务

应用程序服务

本地服务

远程服务

详细分类如下所示。

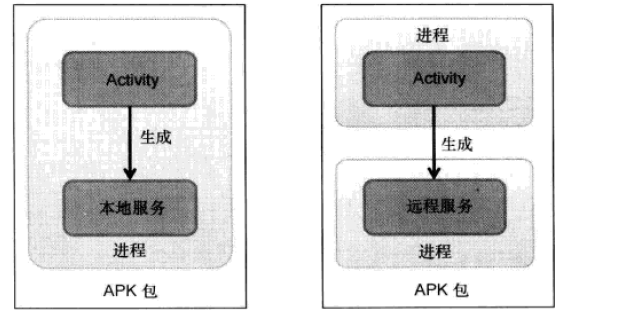


## 11 本地服务和远程服务的区别

本地服务：使用服务的客户端（应用程序）和服务同处于一个进程，当应用程序终止时，服务也终止。

远程服务：使用服务的客户端和服务不在同一进程，客户端终止，服务扔独立运行。

本地服务和远程服务最大不同在于绑定服务时使用的方法不同。



## 12 本地服务中的服务绑定

方法待详细理解

## 13 远程服务中的服务绑定

## 14 andorid中的系统服务

分为本地系统服务和Java平台系统服务。

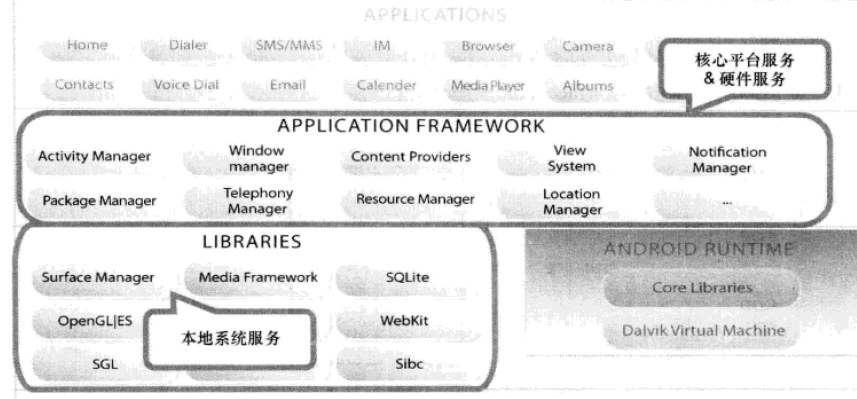
本地系统服务：由c++编写，运行在libraries层。

Java平台服务——核心平台服务

java核心平台服务时支持framework运行的必要服务，如Package Manager Servicem、windowsManagerService 等，位于andorid中的application Framework层。核心平台服务一般不与应用程序直接交互，主要是维持android framework 的正常运行

Java平台服务——硬件服务

硬件服务用于控制物理硬件，为其他服务提供了api。 如，Power Service /WifiService /UsbService等，，位于andorid中的application Framework层。



## 15服务的运行

应用程序服务

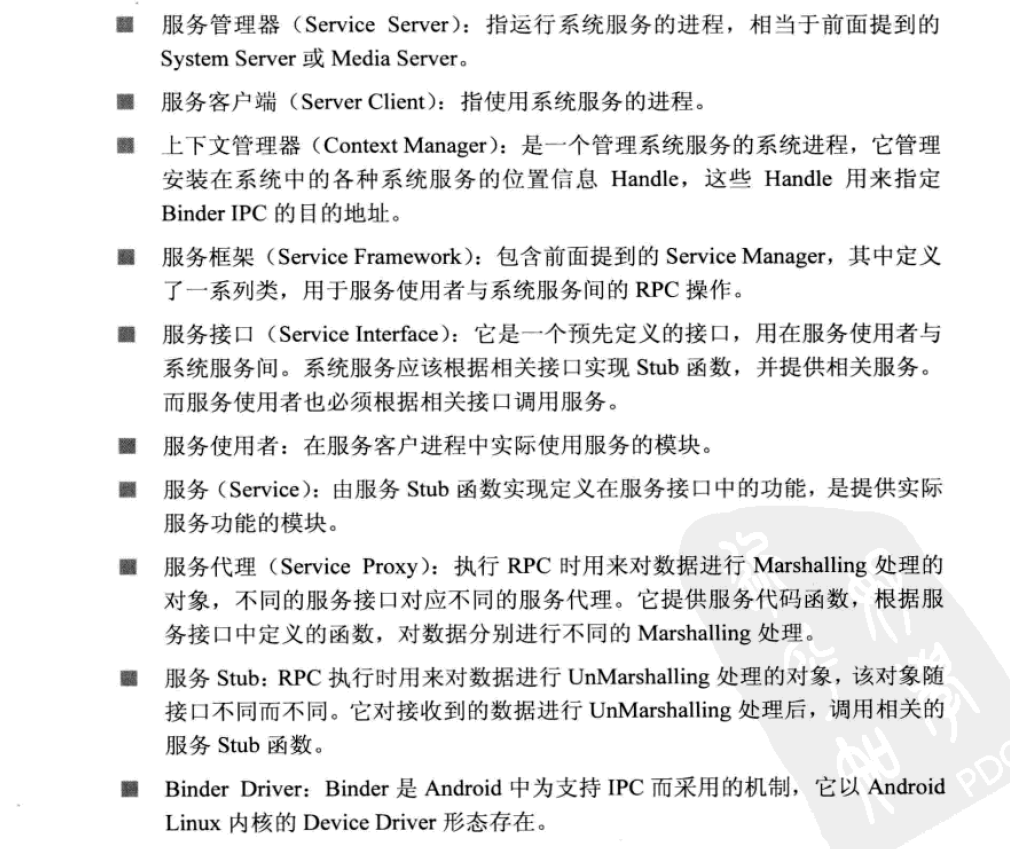
应用程序服务在应用程序启动前，通过调用startServcie（）函数拉起对应的应用程序服务

系统服务

Android中的系统服务都是在系统启动时，由init进程创建和启动。并在后台一直运行，系统关闭时才会停止。因为系统服务是在系统启动时启动的，所以一直运行在后台。在使用系统服务时，只需要调用getSystemService()函数即可。

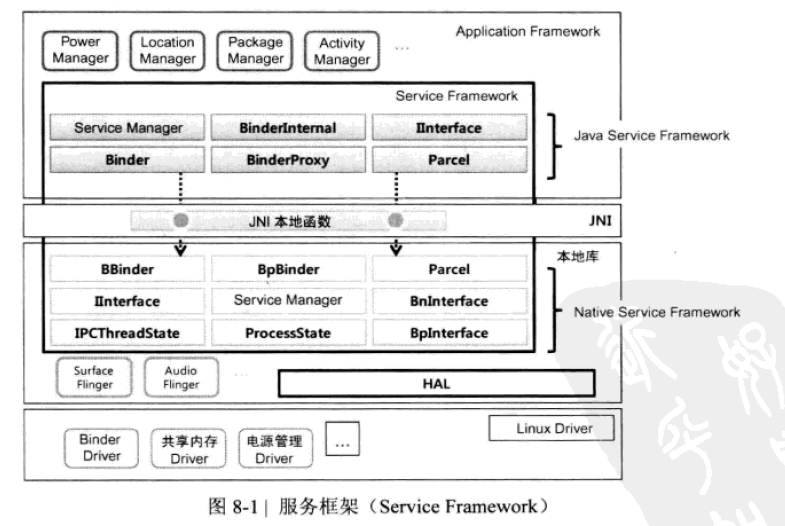
Android启动时，系统服务具体由媒体服务器（media Server）和系统服务器（system Server）两个系统进程运行。

## 16 android中的主要术语



## 17 android中的服务框架

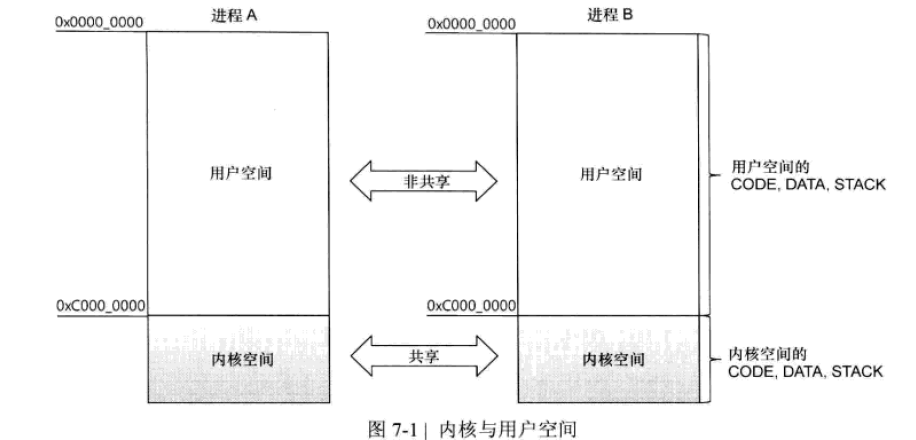
Android中的服务框架是指c++实现的本地服务框架和Java实现的Java服务框架，两层框架之间通过JNI进行 连接。具体如图所示：



## 18 android中的进程空间

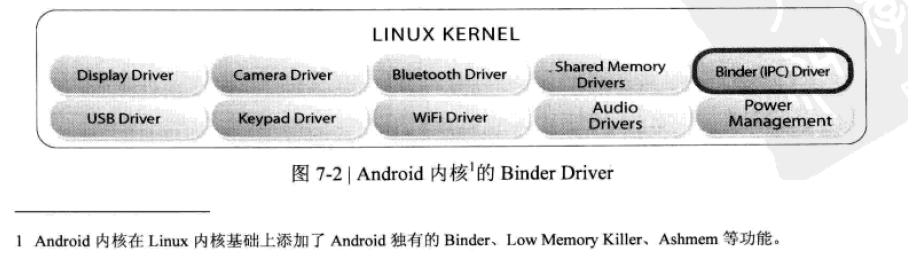
Android进程和linux的进程一样，只运行在固有的虚拟地址空间中，一个4G的虚拟地址空间，3G是用户空间，1G是内核空间。用户的代码和相关的库分别运行在用户空间的代码区域、数据区域以及堆栈区域。

进程拥有独立的用户空间，但是内核空间中的任务数据、代码是共享的。



## 19 Linux的进程间通信和android的Binder通信的区别

Binder既可以实现进程间的通信还可以实现调用另一进程函数（RPC）功能。



## 20 Android的Binder通信。。。。。

在车机系统开发中，base代码中已实现了Binder通信的相关内容，作为开发者，只需按照特定方式使用即可，暂时不需要太关注binder的实现原理。

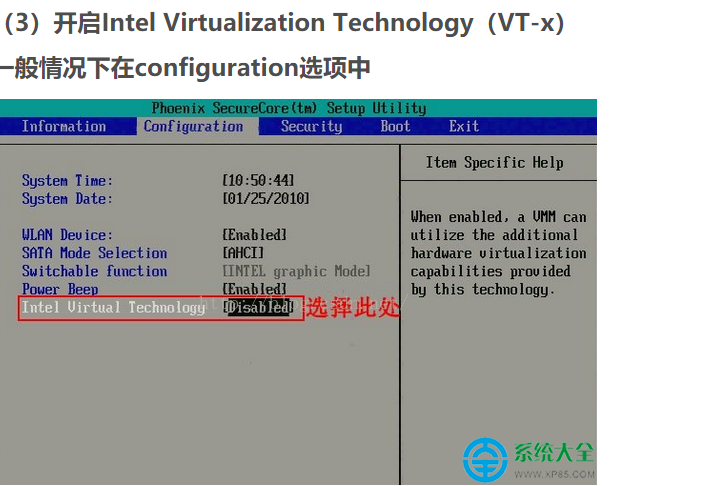
## 21 Android的SDK

Android的sdk中包含了模拟器、教程、api文档和示例代码等。大小4个G左右。

## 22 android Studio安装遇到HAXM installation failed错误

一般是Bois中的Virtualization Technology未开启造成的

进入bois开启

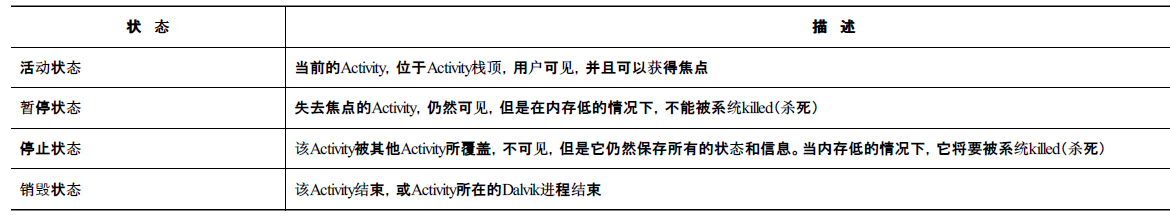




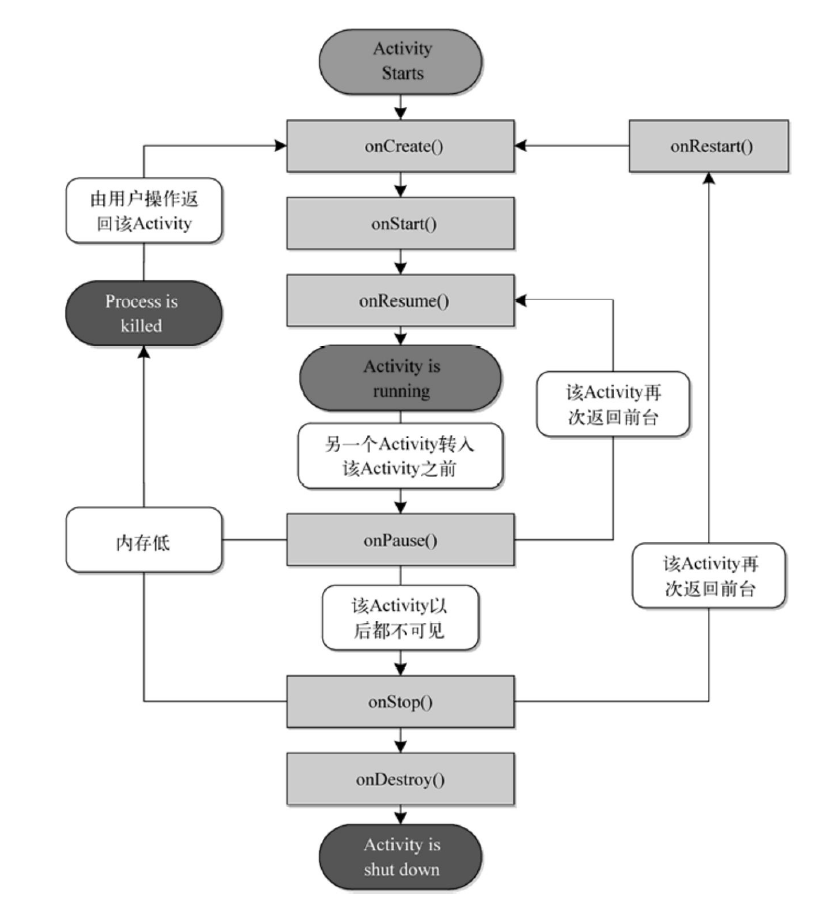
## 23 android四个组件-Activity

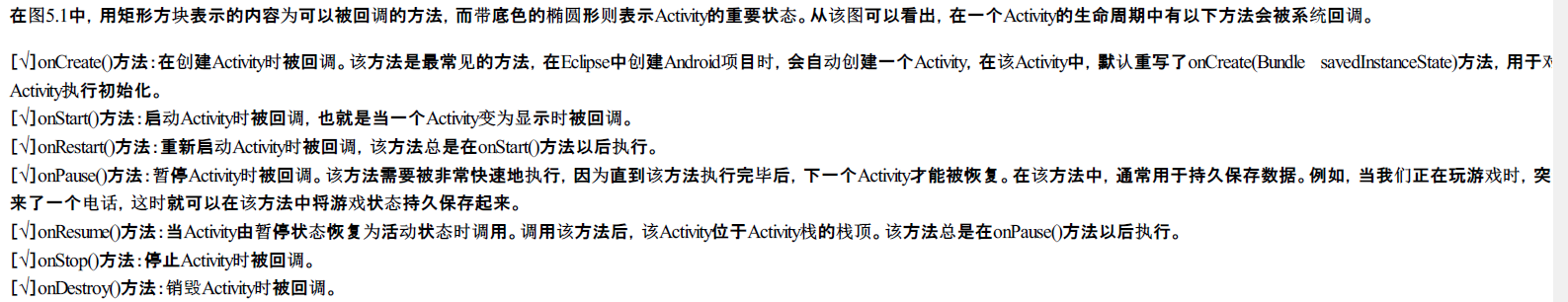
一个activity代表屏幕中的已一屏，activity中可以添加button、text、checkBox等多个组件。一个应用中可以有多个activity，这些activity组成activity栈，当前活动的activity位于栈顶，之前的被压入栈。

### 23.1 activity的状态



### 23.2 activity声明周期





## 24 Intnet

用于android中的各个组件之间的通信。

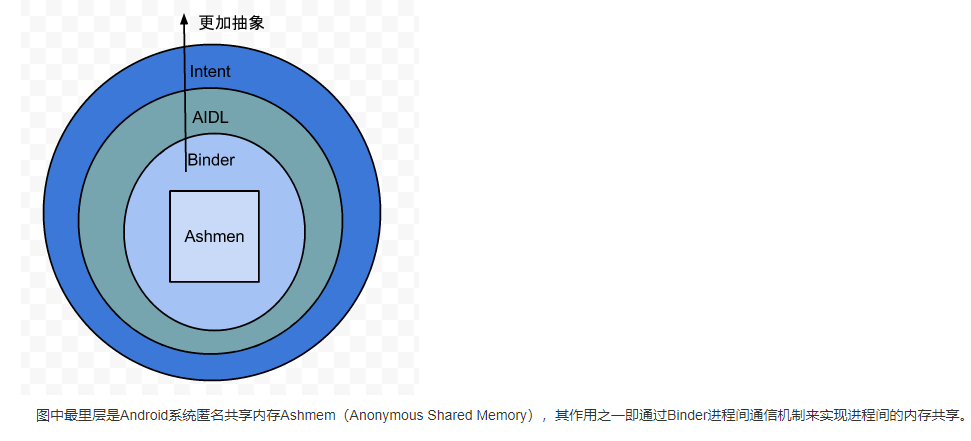
## 25 IPC、Binder、AIDL与Intent之间区别与联系

IPC：Inter-Process Communication(进程间通信)

Binder：Binder进程间通信机制

AIDL：Android Interface Definition Language(android接口定义语言)

Intent：意图



IPC是一种概念，即进程间通信；其它几个都是Android里的概念；

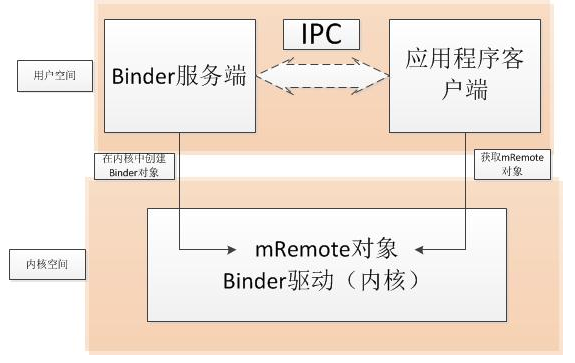
Binder是对IPC的具体实行，是IPC的一种具体实现；

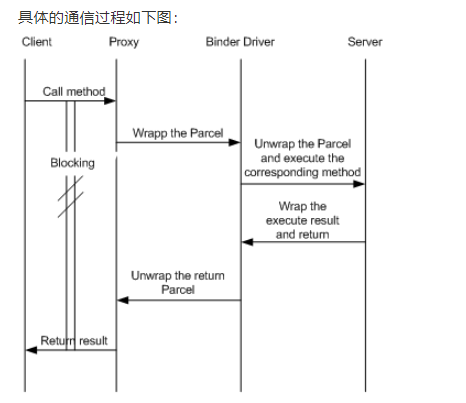
AIDL是Binder机制向外提供的接口，目的就是为了方便对Binder的使用；

Intent是最高层级的封装，实质是封装了对Binder的使用，当然Intent也常常在同一进程中调用，只是把两种方式封装在一起了。

## 26 Binder通信大致过程

Binder通信采用的是client-server通信结构，client与server的通信由Binder驱动程序和Service Manager组件协助，这两个组件Android已经实现好了并由系统运行，而开发者只需要按照框架规范实现client与server接口即可。参考下图：





1.client通过获得一个server的代理接口，对server进行直接调用

2.实际上代理接口中定义的方法与server中定义的方法是一一对应的

3.client调用某个代理接口中的方法时，代理接口的方法会将client传递的参数打包成为Parcel对象

4.代理接口将该Parcel发送给内核中的binder driver.

5.server会读取binder driver中的请求数据，如果是发送给自己的，解包Parcel对象，处理并将结果返回

6.整个的调用过程是一个同步过程，在server处理的时候，client会block住

## 27 为什么使用contentResolver 和contentProvider

数据放在数据库中，对数据的存取只要操作数据库就可以了，为什么还要多此一举使用contentProvider了呢

不同进程间的数据共享问题？ 还是只是单纯的屏蔽数据的存储形式？

1）该篇文章中解释了为什么使用contentProvider ，以及contentProvider和contentResolver的区别

https://www.jianshu.com/p/cdef889736ec

2）牛客网上解释

ContentProvider实现了不同APP之间数据共享，ContentProvider为其他应用程序提供了访问本应用程序的接口，其他应用程序可以通过ContentResolver来操作ContentProvider提供的数据，同时ContentProvider保证了被访数据的安全性，用户只需要关心操作数据的uri就可以了。sql也有增删改查的方法，单sql只能操作本应用下的数据库。