**Android系统启动过程，App启动过程**  
]从桌面点击到activity启动的过程

1. Launcher本质上也是一个应用程序，和我们的App一样，也是继承自Activity, 桌面上的图标，使用的是BubbleTextView对象,bubbletextview的点击事件Launcher.onClick(View v)里面,Launcher线程捕获onclick的点击事件，调用Launcher.startActivitySafely,进一步调用Launcher.startActivity，最后调用父类Activity的startActivity。intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);将 Activity会添加到一个新的Task栈中

2.Activity和ActivityManagerService交互，引入Instrumentation，将启动请求交给Instrumentation，调用Instrumentation.execStartActivity。

3.调用ActivityManagerService的startActivity方法，这里做了进程切换（具体过程请查看源码）。

由Zygote进程孵化出新的应用进程后，会执行ActivityThread类的main方法ActivityThread类的main方法.在该方法里会先准备好Looper和消息队列，然后调用attach方法将应用进程绑定到ActivityManagerService，然后进入loop循环，不断地读取消息队列里的消息，并分发消息。

4. 应用进程接下来通知ActivityManagerService应用进程已启动，ActivityManagerService保存应用进程的一个代理对象，这样ActivityManagerService可以通过这个代理对象控制应用进程，然后ActivityManagerService通知**应用进程**创建入口Activity的实例，并执行它的生命周期方法

**自己总结 app启动过程**

Launcher本质上也是一个应用程序，和我们的App一样，也是继承自Activity, 桌面上的图标，使用的是BubbleTextView对象,bubbletextview的点击事件Launcher.onClick(View v)里面,在onclick方法中挑用startActivitySafely

,挑用launche的startActivity(View v, Intent intent, Object tag),最终挑用Activity.startActivity(intent, opts.toBundle(),使用intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);将 Activity会添加到一个新的Task栈中,最终将app的首次启动页面打开;

**Activity启动过程**

调用startactivity最终调用的都是 startactivityforresult,

* ActivityManagerServices，简称AMS，服务端对象，负责系统中所有Activity的生命周期
* ActivityThread，App的真正入口。当开启App之后，会调用main()开始运行，开启消息循环队列，这就是传说中的UI线程或者叫主线程。与ActivityManagerServices配合，一起完成Activity的管理工作
* ApplicationThread，用来实现ActivityManagerService与ActivityThread之间的交互。在ActivityManagerService需要管理相关Application中的Activity的生命周期时，通过ApplicationThread的代理对象与ActivityThread通讯。
* ApplicationThreadProxy，是ApplicationThread在服务器端的代理，负责和客户端的ApplicationThread通讯。AMS就是通过该代理与ActivityThread进行通信的。
* Instrumentation，每一个应用程序只有一个Instrumentation对象，每个Activity内都有一个对该对象的引用。Instrumentation可以理解为应用进程的管家，ActivityThread要创建或暂停某个Activity时，都需要通过Instrumentation来进行具体的操作。
* ActivityStack，Activity在AMS的栈管理，用来记录已经启动的Activity的先后关系，状态信息等。通过ActivityStack决定是否需要启动新的进程。
* ActivityRecord，ActivityStack的管理对象，每个Activity在AMS对应一个\* ActivityRecord，来记录Activity的状态以及其他的管理信息。其实就是服务器端的Activity对象的映像。
* TaskRecord，AMS抽象出来的一个“任务”的概念，是记录ActivityRecord的栈，一个“Task”包含若干个ActivityRecord。AMS用TaskRecord确保Activity启动和退出的顺序。如果你清楚Activity的4种launchMode，那么对这个概念应该不陌生。

Startactivity

1 使用代理模式启动到ActivityManagerService startactivity中执行；

2 创建记录Activity的状态以及其他的管理信息的ActivityRecord到mHistory记录中；

3 通过socket通信到Zgote相关类创建fock出process；

4 通过ApplicatonThread与ActivityManagerService建立通信；

5 ActivityManagerService通知ActiveThread启动Activity的创建；

6 ActivityThread创建Activity加入到mActivities中并开始调度Activity执行；

<http://www.cnblogs.com/bastard/archive/2012/04/07/2436262.html>

http://blog.csdn.net/u013134722/article/details/60772405#launcher%E6%98%AF%E4%BB%80%E4%B9%88%E4%BB%80%E4%B9%88%E6%97%B6%E5%80%99%E5%90%AF%E5%8A%A8%E7%9A%84