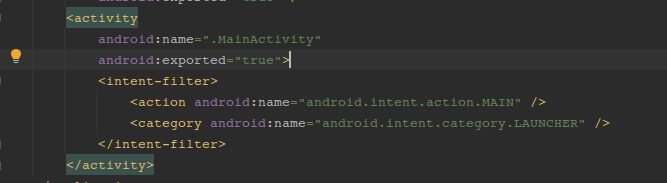
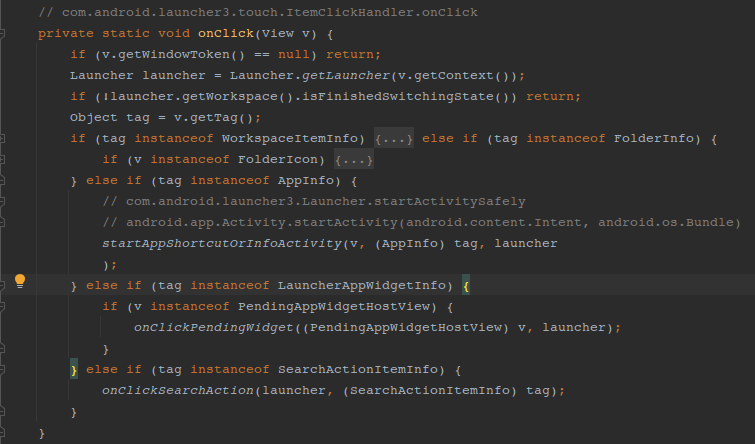
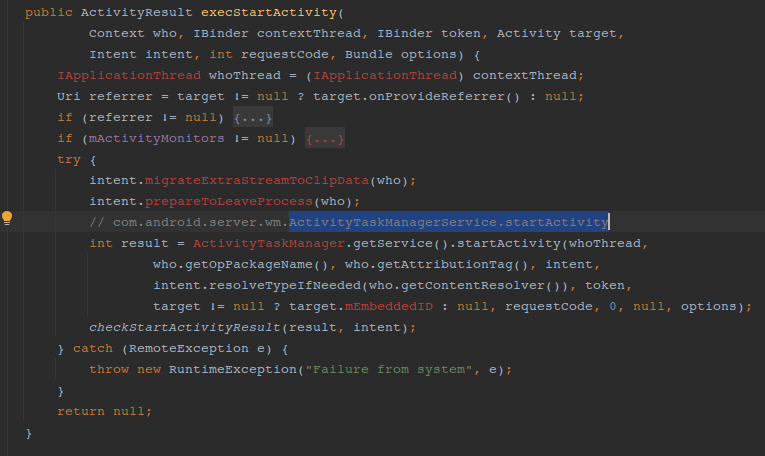
# Activity启动流程

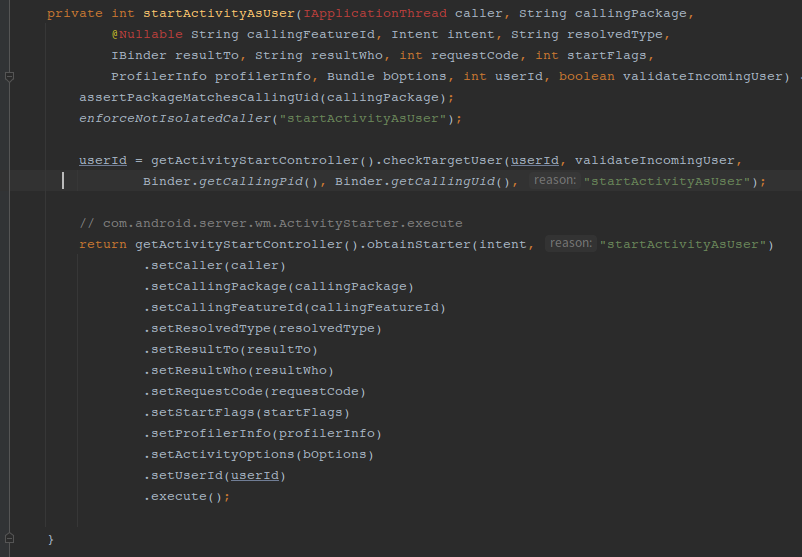
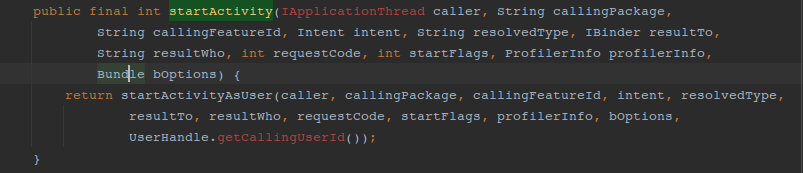
## Launcher点击

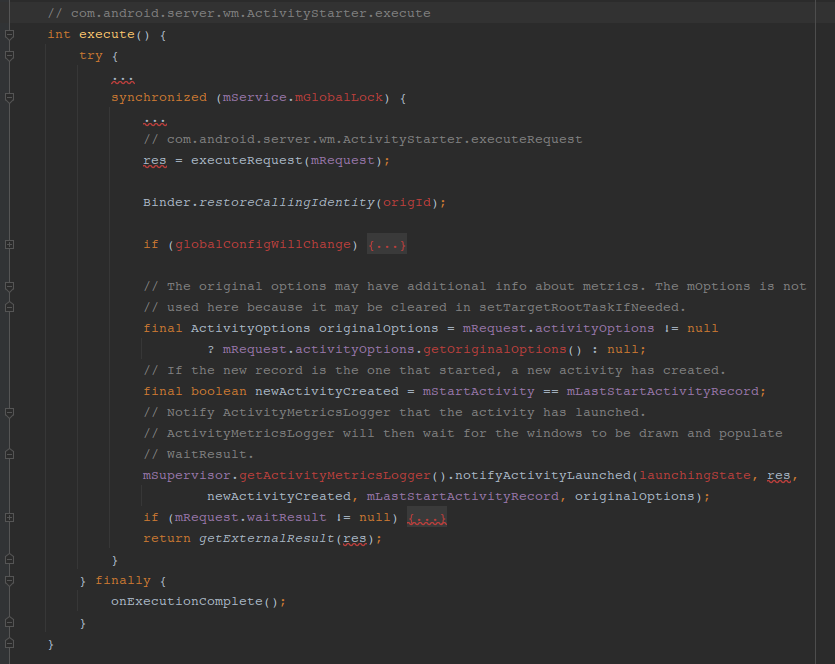
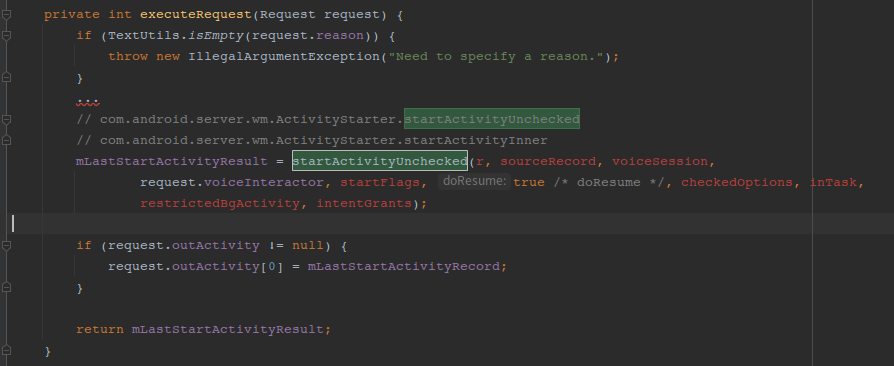
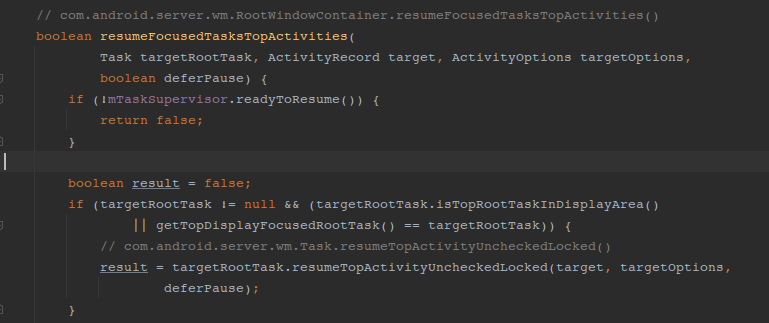
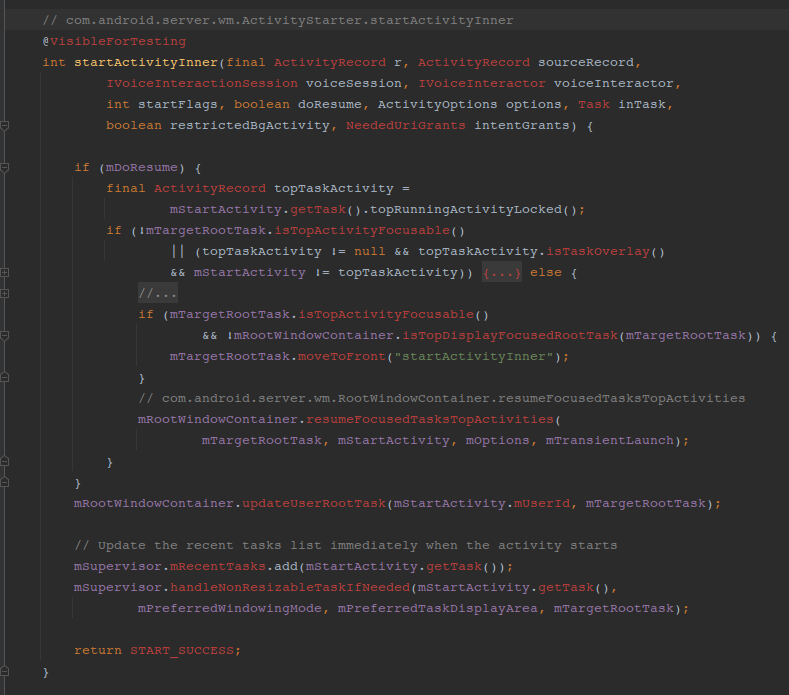
Launcher点击图标启动Activity，携带Intent跨进程调用ActivityTaskManagerService.startActivity





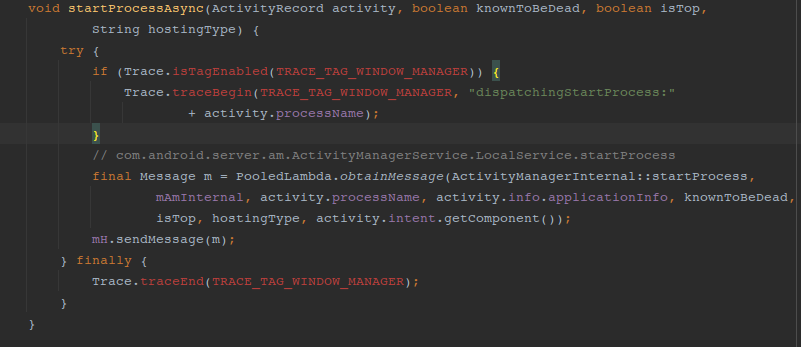
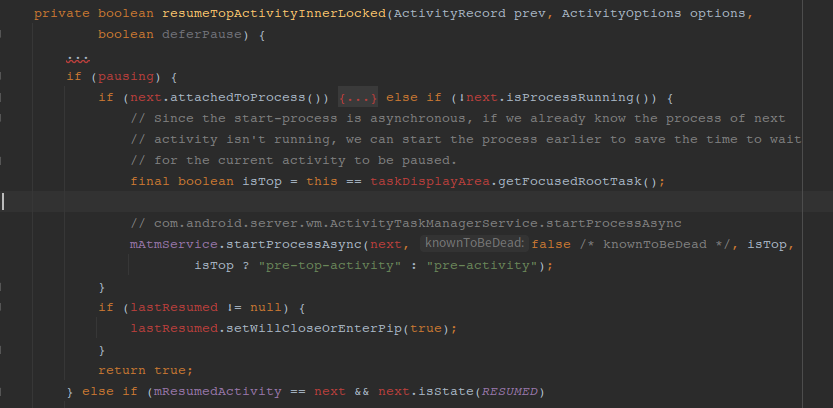
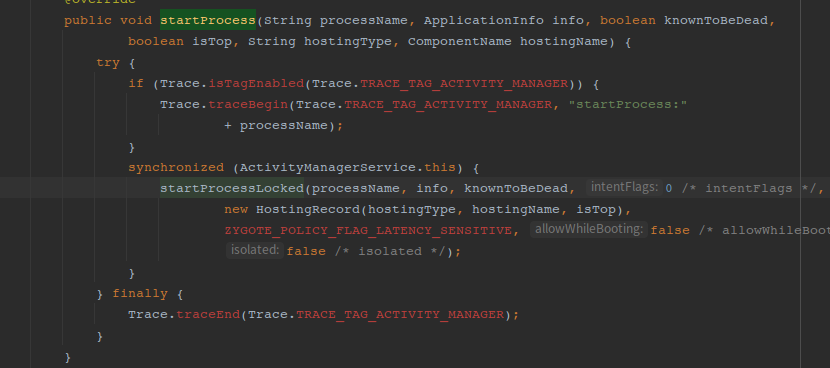
## ActivityTaskMangerService 接受并处理启动请求

 ActivityController 是以委托的方式把启动请求交给了一个包含所有启动信息的 ActivityStarter 对象并调用 execute 方法执行。

  startActivityInner() 方法会根据启动标志位和 Activity 启动模式来决定如何启动一个 Activity 以及是否要调用 deliverNewIntent 方法通知 Activity 有一个 Intent 试图重新启动它，在里面又会调用RootWindowContainer的resumeFocusedTasksTopActivities()方法

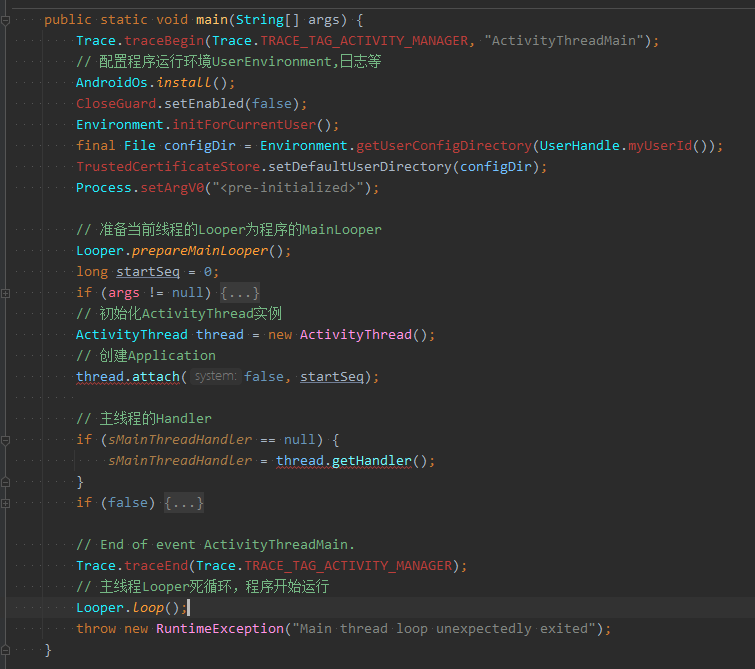
RootWindowContainer的resumeFocusedTasksTopActivities()方法会调用

Task的resumeTopActivityUncheckedLocked()方法，接着又调用resumeTopActivityInnerLocked()方法，resumeTopActivityInnerLocked() 方法中会先判断是否由 Activity 处于 Resumed 状态，如果有的话先让这个 Activity 执行 pause 过程，

然后这个方法会根据进程和线程是否为空判断 App 是否已经启动，如果没有启动则会调用ActivityTaskManagerService的startProcessAsync方法创建新的进程。 通过Handler消息传递调用ActivityManagerService.LocalService#startProcess()方法，然后经过多次调用通过 Process.start()

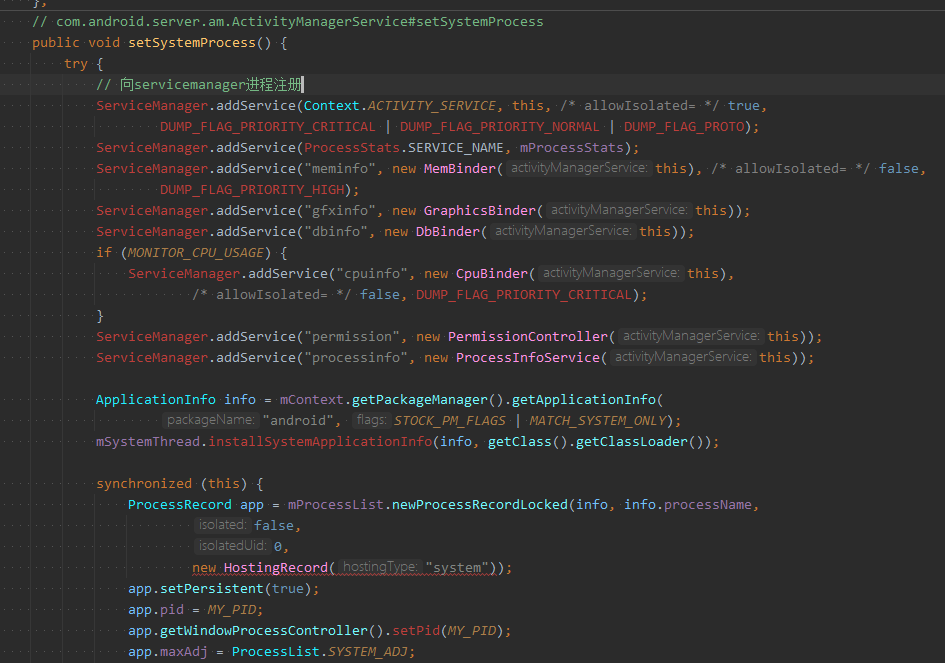
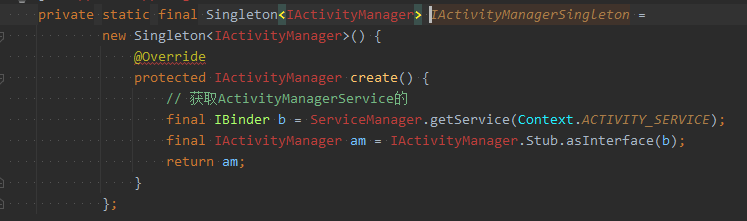
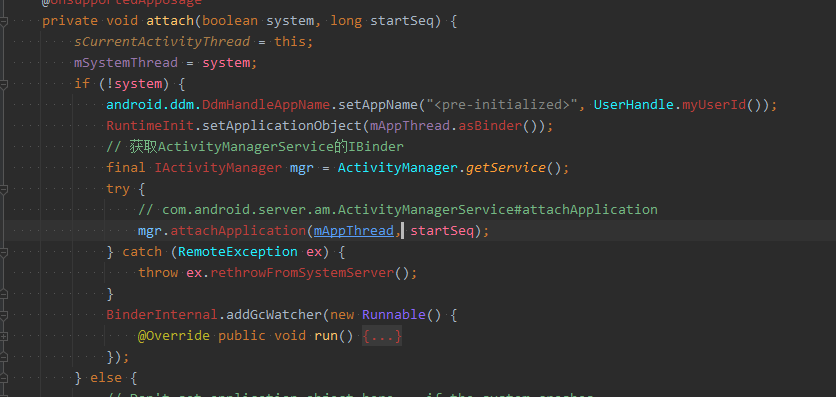
最终是调用 Zygote 进程并通过 socket 通信的方式让 Zygote 进程 fork 出一个新的进程，并通过反射创建 ActivityThread 主线程并对其进行初始化。现在，要启动的 Activity 的进程和主线程皆已创建完毕。在 ActivityThread.attach 方法中，首先会通过 AMS 为新应用绑定一个 Application。

## 从android.app.ActivityThread#main()开始



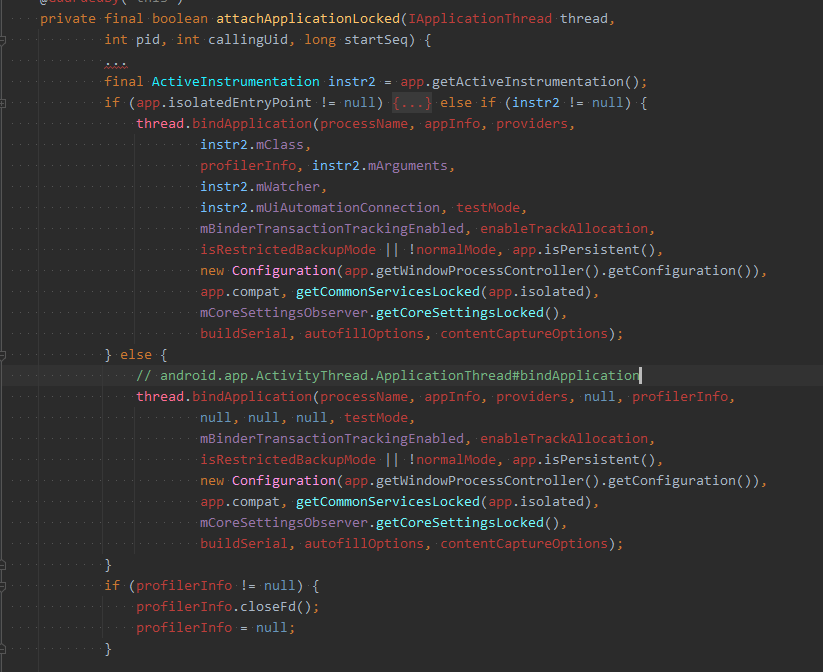
1. 配置程序运行环境UserEnvironment,日志等。
2. 准备当前线程的Looper为程序的MainLooper。
3. 初始化ActivityThread实例
4. ActivityThread实例调用attach(false)，创建Application
5. 获取MainThreadHandler。
6. MainLooper开始loop(),接收发送消息。程序开始运行

## 创建Application消息

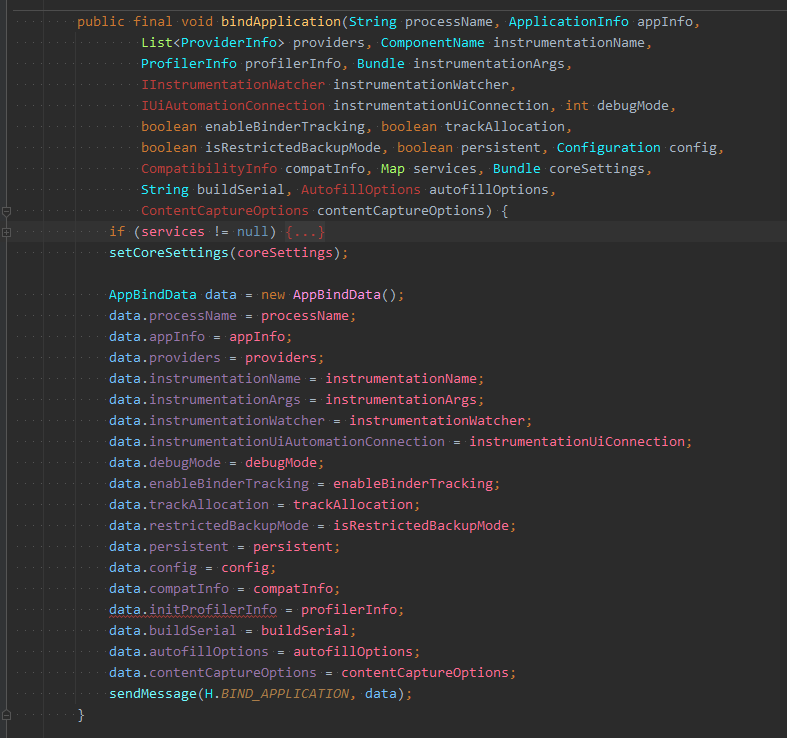
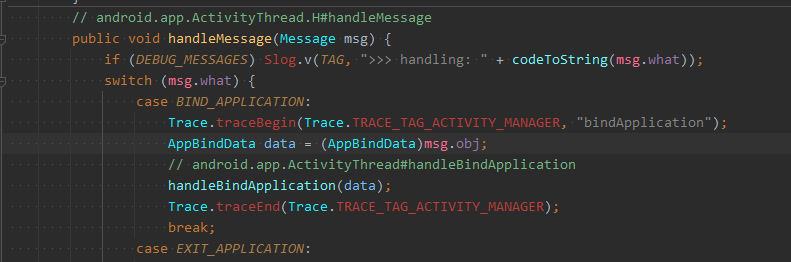
1. 判断是否系统程序，不同的初始化流程(分析非系统)
2. 获得ActivityManager实例——ActivityManagerService，ActivityManager.getService()通过ServiceManager获得ActivityManagerService的IBinder，目的既是为了通过这个IBinder和ActivityManagerService进行通讯。
3. mAppThread
   1. final常量，不允许修改
   2. ApplicationThread是ActivityThread内部类
   3. IapplicationThread的服务端
   4. 承担发送Activity生命周期、及其它一些任务。ApplicationThread传到ActivityManager中为了让SystemServer根据情况与应用通讯。

## ActivityManagerService调度发送初始化消息

ApplicationThread以IApplicationThread的身份到了ActivityManagerService中，经过一系列的操作，最终被调用了自己的bindApplication()方法，发出初始化Applicationd的消息，发了一条**H.BIND\_APPLICATION**消息，程序开始。



## ApplicationThread收到初始化消息

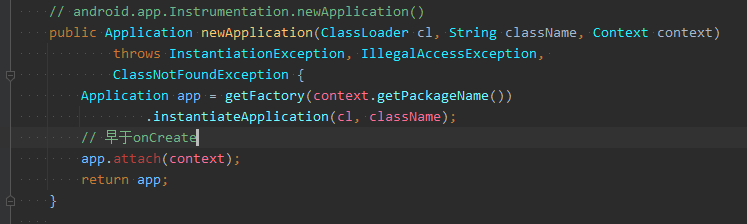


## Instrumentation

1. Instrumentation会在应用程序的任何代码运行之前被实例化，它能够允许你监视应用程序和系统的所有交互。
2. 收集AndroidManifest.xml标签信息。
3. Apllication的创建，Activity的创建，以及生命周期都会经过这个对象去执行。简单点说，就是把这些操作包装了一层。通过操作Instrumentation进而实现上述的功能。

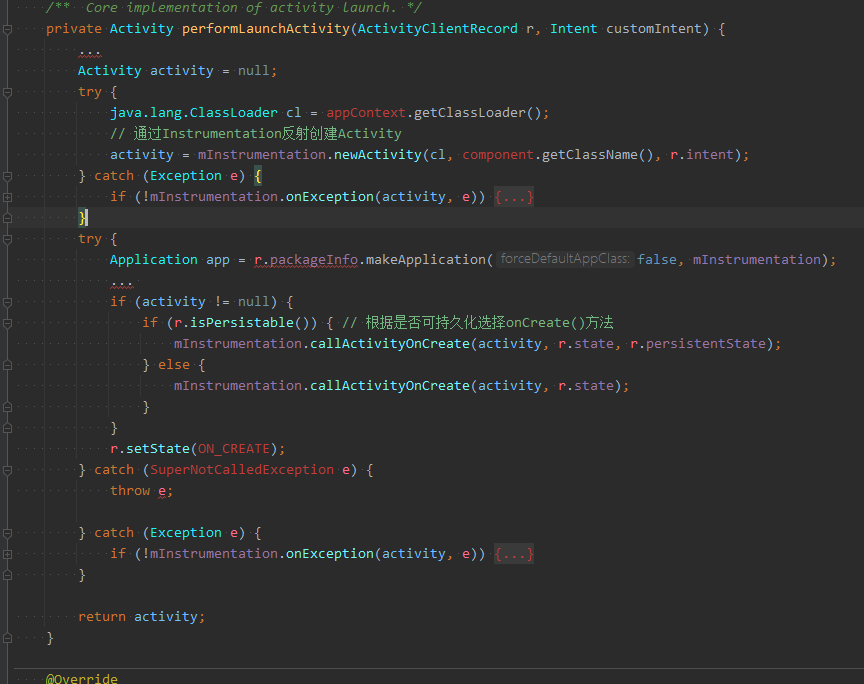
## LoadedApk就是data.info

在取得Application的实际类名之后，最终的创建工作还是交由Instrumentation去完成

## LaunchActivity

当Application初始化完成后，系统会根据Manifests中的配置启动Activity发送一个Intent去启动相对应的Activity。H就收到了一条LAUNCH\_ACTIVITY的消息，收到消息后，真正处理是在ActivityThread中的handleLaunchActivity()中进行的

**Application 创建完成，第一个Activity创建完成。**