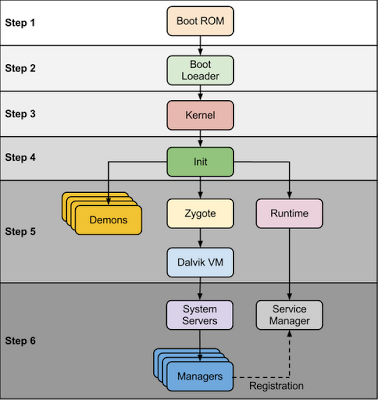
**Android启动过程深入解析**

* 当按下Android设备电源键时究竟发生了什么？
* Android的启动过程是怎么样的？
* 什么是Linux内核？
* 桌面系统linux内核与Android系统linux内核有什么区别？
* 什么是引导装载程序？
* 什么是Zygote？
* 什么是X86以及ARM linux？
* 什么是init.rc?
* 什么是系统服务？

当我们想到Android启动过程时，脑海中总是冒出很多疑问。本文将介绍Android的启动过程，希望能帮助你找到上面这些问题的答案。

Android是一个基于Linux的开源操作系统。x86（x86是一系列的基于intel 8086 CPU的计算机微处理器指令集架构）是linux内核部署最常见的系统。然而，所有的Android设备都是运行在ARM处理器（ARM 源自进阶精简指令集机器，源自ARM架构）上，除了英特尔的Xolo设备(http://xolo.in/xolo-x900-features)。 Xolo来源自凌动1.6GHz x86处理器。Android设备或者嵌入设备或者基于linux的ARM设备的启动过程与桌面版本相比稍微有些差别。这篇文章中，我将解释 Android设备的启动过程。[深入linux启动过程](http://www.ibm.com/developerworks/linux/library/l-linuxboot/)是一篇讲桌面linux启动过程的好文。

当你按下电源开关后Android设备执行了以下步骤。



此处图片中step2中的一个单词拼写错了，Boot Loaeder应该为Boot Loader（多谢@jameslast 提醒）

### 第一步：启动电源以及系统启动

当电源按下，引导芯片代码开始从预定义的地方（固化在ROM）开始执行。加载引导程序到RAM，然后执行。

### 第二步：引导程序

引导程序是在Android操作系统开始运行前的一个小程序。引导程序是运行的第一个程序，因此它是针对特定的主板与芯片的。设备制造商要么使用很受欢迎的引导程序比如[redboot](http://ecos.sourceware.org/redboot/)、[uboot](http://www.denx.de/wiki/U-Boot)、[qi bootloader](http://wiki.openmoko.org/wiki/Qi)或者开发自己的引导程序，它不是Android操作系统的一部分。引导程序是OEM厂商或者运营商加锁和限制的地方。

引导程序分两个阶段执行。第一个阶段，检测外部的RAM以及加载对第二阶段有用的程序；第二阶段，引导程序设置网络、内存等等。这些对于运行内核是必要的，为了达到特殊的目标，引导程序可以根据配置参数或者输入数据设置内核。

Android引导程序可以在\bootable\bootloader\legacy\usbloader找到。  
传统的加载器包含的个文件，需要在这里说明：

1. init.s初始化堆栈，清零BBS段，调用main.c的\_main()函数；
2. main.c初始化硬件（闹钟、主板、键盘、控制台），创建linux标签。

更多关于Android引导程序的可以在[这里](https://motorola-global-portal.custhelp.com/app/answers/detail/a_id/86208/%7E/bootloader-frequently-asked-questions)了解。

### 第三步：内核

Android内核与桌面linux内核启动的方式差不多。内核启动时，设置缓存、被保护存储器、计划列表，加载驱动。当内核完成系统设置，它首先在系统文件中寻找”init”文件，然后启动root进程或者系统的第一个进程。

### 第四步：init进程

init是第一个进程，我们可以说它是root进程或者说有进程的父进程。init进程有两个责任，一是挂载目录，比如/sys、/dev、/proc，二是运行init.rc脚本。

* init进程可以在/system/core/init找到。
* init.rc文件可以在/system/core/rootdir/init.rc找到。
* readme.txt可以在/system/core/init/readme.txt找到。

对于init.rc文件，Android中有特定的格式以及规则。在Android中，我们叫做Android初始化语言。  
Android初始化语言由四大类型的声明组成，即Actions（动作）、Commands（命令）、Services（服务）、以及Options（选项）。  
Action（动作）：动作是以命令流程命名的，有一个触发器决定动作是否发生。  
语法

|  |  |
| --- | --- |
| 1  2  3  4 | on <trigger>      <command>      <command>      <command> |

Service（服务）：服务是init进程启动的程序、当服务退出时init进程会视情况重启服务。  
语法

|  |  |
| --- | --- |
| 1  2  3  4 | service <name> <pathname> [<argument>]\*      <option>      <option>      ... |

Options（选项）  
选项是对服务的描述。它们影响init进程如何以及何时启动服务。  
咱们来看看默认的init.rc文件。这里我只列出了主要的事件以及服务。  
Table

|  |  |
| --- | --- |
| Action/Service | 描述 |
| on early-init | 设置init进程以及它创建的子进程的优先级，设置init进程的安全环境 |
| on init | 设置全局环境，为cpu accounting创建cgroup(资源控制)挂载点 |
| on fs | 挂载mtd分区 |
| on post-fs | 改变系统目录的访问权限 |
| on post-fs-data | 改变/data目录以及它的子目录的访问权限 |
| on boot | 基本网络的初始化，内存管理等等 |
| service servicemanager | 启动系统管理器管理所有的本地服务，比如位置、音频、Shared preference等等… |
| service zygote | 启动zygote作为应用进程 |

在这个阶段你可以在设备的屏幕上看到“Android”logo了。

### 第五步

在Java中，我们知道不同的虚拟机实例会为不同的应用分配不同的内存。假如Android应用应该尽可能快地启动，但如果Android系统为每 一个应用启动不同的Dalvik虚拟机实例，就会消耗大量的内存以及时间。因此，为了克服这个问题，Android系统创造了”Zygote”。 Zygote让Dalvik虚拟机共享代码、低内存占用以及最小的启动时间成为可能。Zygote是一个虚拟器进程，正如我们在前一个步骤所说的在系统引 导的时候启动。Zygote预加载以及初始化核心库类。通常，这些核心类一般是只读的，也是Android SDK或者核心框架的一部分。在Java虚拟机中，每一个实例都有它自己的核心库类文件和堆对象的拷贝。

Zygote加载进程

1. 加载[ZygoteInit](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/2.2_r1.1/com/android/internal/os/ZygoteInit.java)类，源代码：/frameworks/base/core/java/com/android/internal/os/ZygoteInit.java
2. registerZygoteSocket()为zygote命令连接注册一个服务器套接字。
3. preloadClassed “preloaded-classes”是一个简单的包含一系列需要预加载类的文本文件，你可以在<Android Source>/frameworks/base找到“preloaded-classes”文件。
4. preloadResources() preloadResources也意味着本地主题、布局以及android.R文件中包含的所有东西都会用这个方法加载。

在这个阶段，你可以看到启动动画。

### 第六步：系统服务或服务

完成了上面几步之后，运行环境请求Zygote运行系统服务。系统服务同时使用native以及java编写，系统服务可以认为是一个进程。同一个 系统服务在Android SDK可以以System Services形式获得。系统服务包含了所有的System Services。

Zygote创建新的进程去启动系统服务。你可以在ZygoteInit类的”startSystemServer”方法中找到源代码。

核心服务：

1. 启动电源管理器；
2. 创建Activity管理器；
3. 启动电话注册；
4. 启动包管理器；
5. 设置Activity管理服务为系统进程；
6. 启动上下文管理器；
7. 启动系统Context Providers；
8. 启动电池服务；
9. 启动定时管理器；
10. 启动传感服务；
11. 启动窗口管理器；
12. 启动蓝牙服务；
13. 启动挂载服务。

其他服务：

1. 启动状态栏服务；
2. 启动硬件服务；
3. 启动网络状态服务；
4. 启动网络连接服务；
5. 启动通知管理器；
6. 启动设备存储监视服务；
7. 启动定位管理器；
8. 启动搜索服务；
9. 启动剪切板服务；
10. 启动登记服务；
11. 启动壁纸服务；
12. 启动音频服务；
13. 启动耳机监听；
14. 启动AdbSettingsObserver（处理adb命令）。

### 第七步：引导完成

一旦系统服务在内存中跑起来了，Android就完成了引导过程。在这个时候“ACTION\_BOOT\_COMPLETED”开机启动广播就会发出去。