# Android 启动过程详解

Android 从 Linux 系统启动有4个步骤；

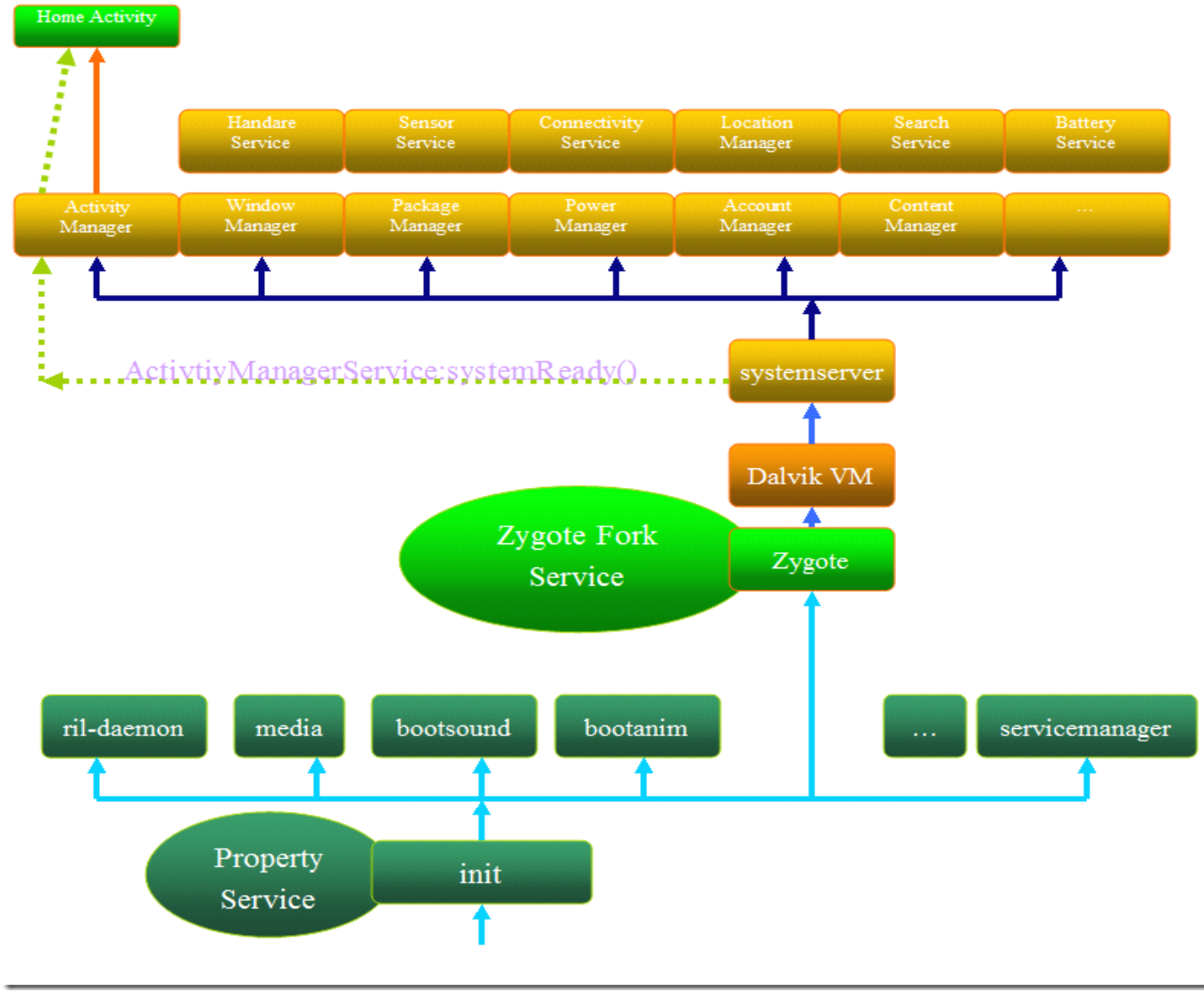
(1)init 进程启动

(2)Native 服务启动

(3)System Server，Android 服务启动

(4)Home 启动

总体启动框架图如：

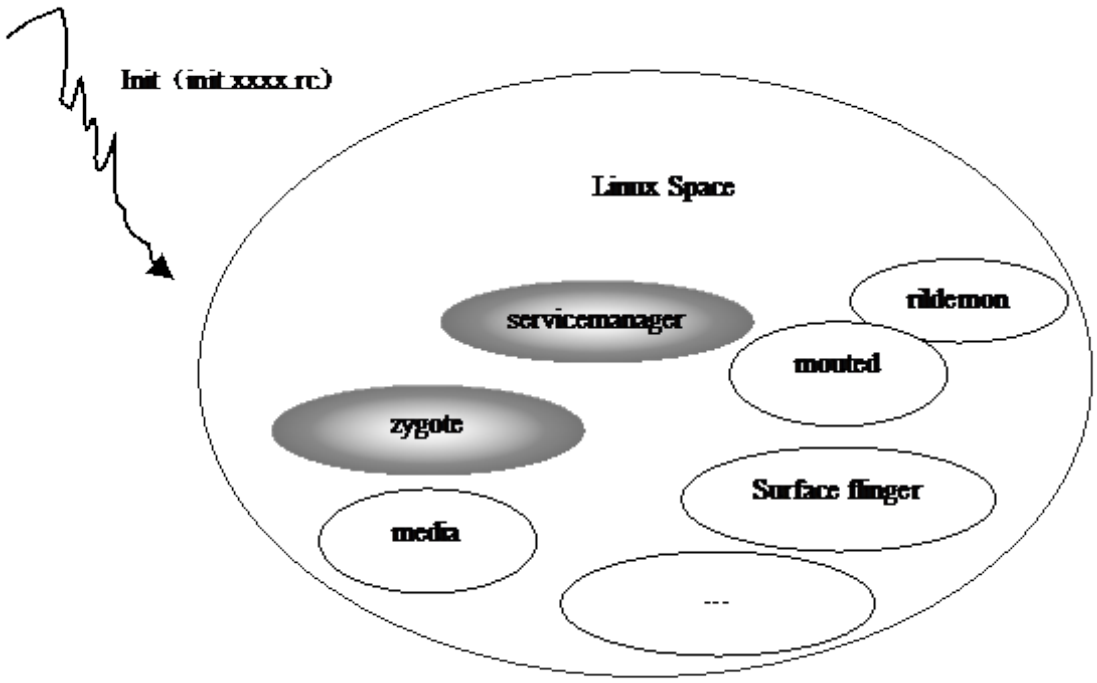


## 第一步：initial 进程(system/core/init)

init 进程，它是一个由内核启动的用户级进程。内核自行启动（已经被载入内存，开始运行，并已初始化所有的设备驱动程序和数据结构等）之后，就通过启动一个用户级程序 init 的方式，完成引导进程。init 始终是第一个进程.

Init.rc

Init.marvell.rc




Init 进程一起来就根据 init.rc 和 init.xxx.rc 脚本文件建立了几个基本的服务：

servicemanamger

zygote

。。。

最后 Init 并不退出，而是担当起 property service 的功能。

## 1.1脚本文件

init@System/Core/Init

Init.c： parse_config_file(Init.rc)

@parse_config_file(Init.marvel.rc)

解析脚本文件：Init.rc 和 Init.xxxx.rc(硬件平台相关)

Init.rc 是 Android 自己规定的初始化脚本(Android Init Language, System/Core/Init/readme.txt)

该脚本包含四个类型的声明：

Actions

Commands

Services

Options.

## 1.2 服务启动机制

我们来看看 Init 是这样解析.rc 文件开启服务的。

（1）打开.rc 文件，解析文件内容@ system/core/init/init.c

将 service 信息放置到 service_list 中。@ system/core/init parser.c

（2）restart_service()@ system/core/init/init.c

service_start

execve(...).建立 service 进程。

## 第二步 Zygote

Servicemanager 和 zygote 进程就奠定了 Android 的基础。Zygote 这个进程起来才会建立起真正的 Android 运行空间，初始化建立的 Service 都是 Navtive service.在.rc 脚本文件中 zygote 的描述：

service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server

所以 Zygote 从 main(...)@frameworks/base/cmds/app_main.cpp 开始。

(1) main(...)@frameworks/base/cmds/app_main.cpp

建立 Java Runtime

runtime.start("com.android.internal.os.ZygoteInit",startSystemServer);

(2)runtime.start@AndroidRuntime.cpp

建立虚拟机

运行：com.android.internal.os.ZygoteInit：main 函数。

（3）main()@com.android.internal.os.ZygoteInit//正真的 Zygote。

registerZygoteSocket();//登记 Listen 端口

startSystemServer();

进入 Zygote 服务框架。

经过这几个步骤，Zygote 就建立好了，利用 Socket 通讯，接收

ActivityManangerService 的请求，Fork 应用程序。

## 第三步 System Server

startSystemServer@com.android.internal.os.ZygoteInit 在 Zygote 上 fork 了一个进程: com.android.server.SystemServer.于是 SystemServer@(SystemServer.java) 就建立了。Android 的所有服务循环框架都是建立 SystemServer@(SystemServer.java) 上。在 SystemServer.java 中看不到循环结构，只是可以看到建立了 init2 的实现函数，建立了一大堆服务，并 AddService 到 service Manager。

```
main() @ com/android/server/SystemServer
{
init1();
}
```

Init1()是在 Native 空间实现的（com_andoird_server_systemServer.cpp)。我们一看这个函数就知道了，init1->system_init() @System_init.cpp

在 system_init()我们看到了循环闭合管理框架。

```
{
Call "com/android/server/SystemServer", "init2"
…..
ProcessState::self()->startThreadPool();
```

```
    IPCThreadState::self()->joinThreadPool();
}
```

init2()@SystemServer.java 中建立了 Android 中所有要用到的服务。

这个 init2（）建立了一个线程，来 New Service 和 AddService 来建立服务

**第三步 Home 启动**

在 ServerThread@SystemServer.java 后半段，我们可以看到系统在启动完所有的 Android 服务后，做了这样一些动作：

（1）使用 xxx.systemReady()通知各个服务，系统已经就绪。

(2) 特别对于 ActivityManagerService.systemReady(回调) Widget.wallpaper,imm(输入法)等 ready 通知。

Home 就是在 ActivityManagerService.systemReady()通知的过程中建立的。下面是 ActivityManagerService.systemReady()的伪代码：

systemReady()@ActivityManagerService.java

resumeTopActivityLocked()

startHomeActivityLocked();//如果是第一个则启动 HomeActivity。

startActivityLocked（。。。）CATEGORY_HOME