Android的启动—init进程

**前言**

通过本篇，我们应理解以下知识点：

* Android系统的启动顺序
* init进程的处理及rc配置文件
* 系统级service是如何启动的
* 添加一个系统级service的注意点

本文主要以android-7.1.2\_r11为参考。

**Android系统启动过程**

Android系统的启动，主要经过如下三个过程：

1. **启动bootloader**

（Bootloader从字面上来看就是启动加载的意思相当于window系统中的bios,用于根据基带初始化硬件）

Boot loader的源码路径：bootable/bootloader/\*。

在系统加电后，CPU将先执行bootloader

1. **启动kernel**

操作系统内核，lunux内核

Kernel源码路径：kernel/\*

Kernel是由boot loader启动。

**3.  建立Android runtime运行环境(以下都属于创建Android runtime运行环境)**

a)   启动Init进程

b)   启动Zygote进程

c)   创建ART虚拟机

d)   启动System server

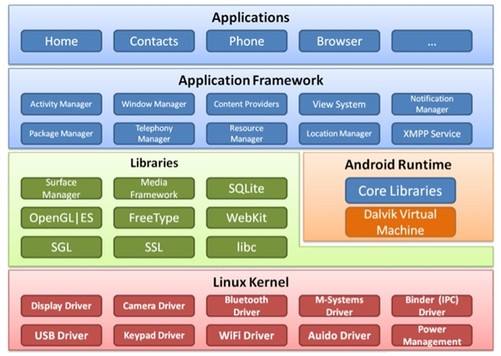
e)   启动其他Android系统级服务

f)   启动桌面Launcher服务

关于boot loader和内核kernel的启动，在此不做深究。

本篇主要探讨Android init进程的处理过程及zygote进程的启动过程。

有关Init.rc配置文件结构及其解析过程，请关注本博客后续专题。



**Init进程的处理**

Init进程是在Kernel启动后而创建的第一个进程。它是一个root进程，是其他进程的父进程。

Init源码路径：system\core\init\。

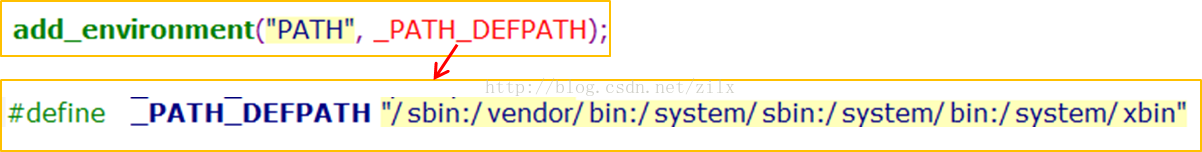
在Init启动后会调用system\core\init\Init.cpp中的main()函数进行处理。从处理中可以看到，init进程主要做了如下两件事：

1. 搭建系统运行环境，创建相关目录，设置相关路径和属性。

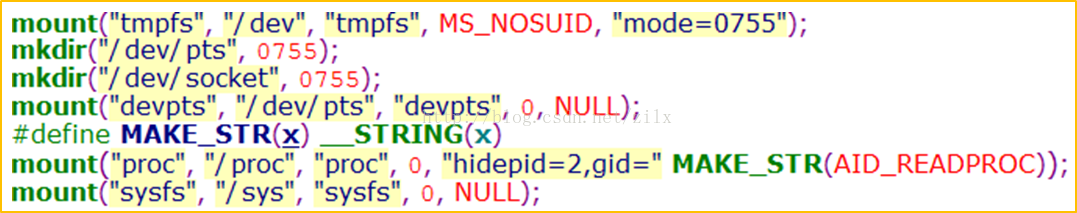
2. 解析rc配置文件，执行rc配置文件所要求的动作和命令。

下面是Init.cpp的main（）函数主要处理的功能，以帮助理解init进程的处理。

**1. 添加相关程序库路径到当前环境**

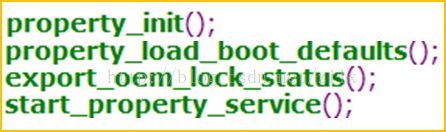


**2.  创建相应目录，并把kernel中创建好的文件系统mount到对应目录上**



**3. 创建并打开init进程的标准输入、输出、出错设备节点和kmsg设备节点**

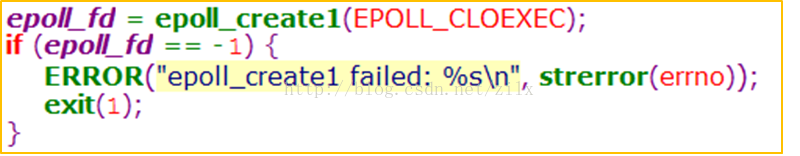
**4.  初始化property，设置系统属性，创建对应socket并监听。Property的初始化和处理是在这里进行的**



**5.  初始化SELinux安全子系统**



**6.  创建epoll句柄**



**7.  初始化signal handler**



**8.  读取并解析rc配置文件。非常重要的处理，其处理内容对用户自定义系统级Service时的配置及设定处理有很大帮助，应加以深入理解**



**9.  根据rc文件配置，one by one的执行所有配置的action，其中就包括启动相关服务。**



**Init进程是如何启动其他service进程的**

在Init.cpp的main（）函数中，有一个重要的功能，就是读取rc配置文件并解析，然后执行配置相关的命令。其中对于Service，我们需要清楚Init进程是如何进行启动的。下面就以Zygote进程的启动为例进行剖析。

从Init.cpp main（）函数处理rc配置文件的过程可知，rc配置文件是一切Action的源，所以要了解Init进程是如何启动Service的，就需要理解rc配置中的各个字段的含义。

大家都知道，Zygote进程是通过Init进程创建的，而且Zygote服务是在init.rc中import对应的zygote配置文件进行配置的。如下图所示：



在init.zygote32.rc中，对zygote服务进行配置：

上述配置就是一个Service section。从Parser处理可知，上述配置只是创建了一个Service对象，并对Service对象进行参数设置，同时把该对象添加到Service 向量表里。此时该Service并未被启动。

那么，zygote service是在什么时候启动的呢？是由谁控制启动的呢？

咱们再回到init.rc配置文件。在rc配置文件中，主要由三个关键字组成若干个section，这三个关键字是：

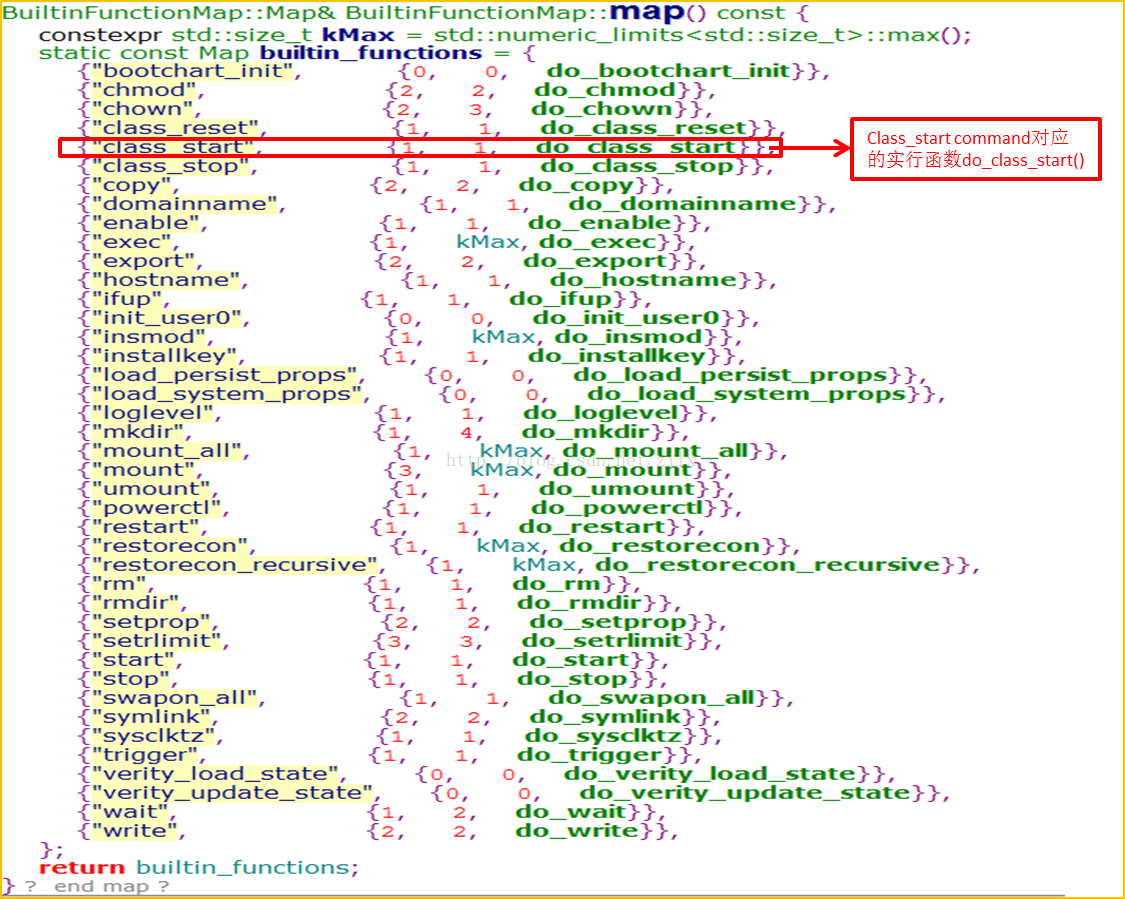
|  |  |
| --- | --- |
| Import | 导入其他配置文件。 |
| service | 配置Service。在解析中创建Service对象，并根据配置设置相应参数。 |
| on | Action配置，具体的command操作。 |

所以如果要启动zygote service，那么就需要在rc中的on字段配置相应的action。

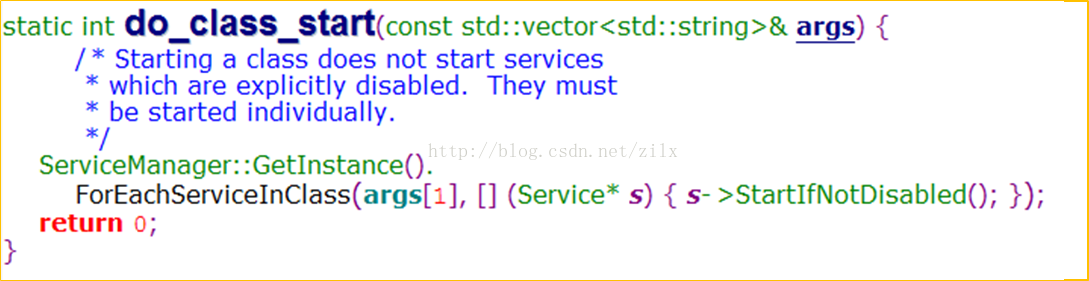
事实正如所想的一样，zygote service的启动是由init.rc中的如下Action处理的，class\_start command就是用来启动service的：

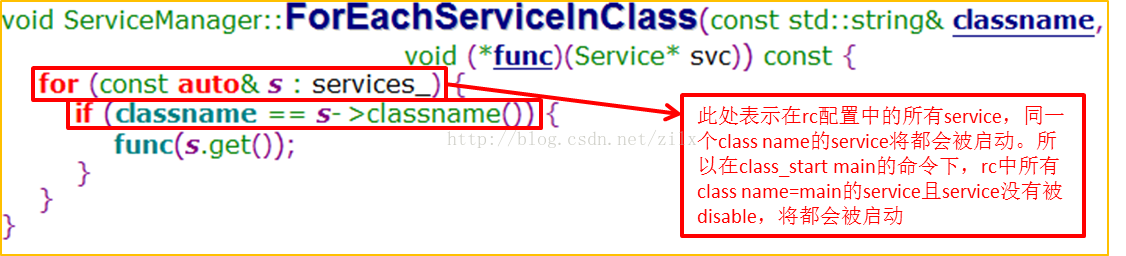


Class\_start command的处理：

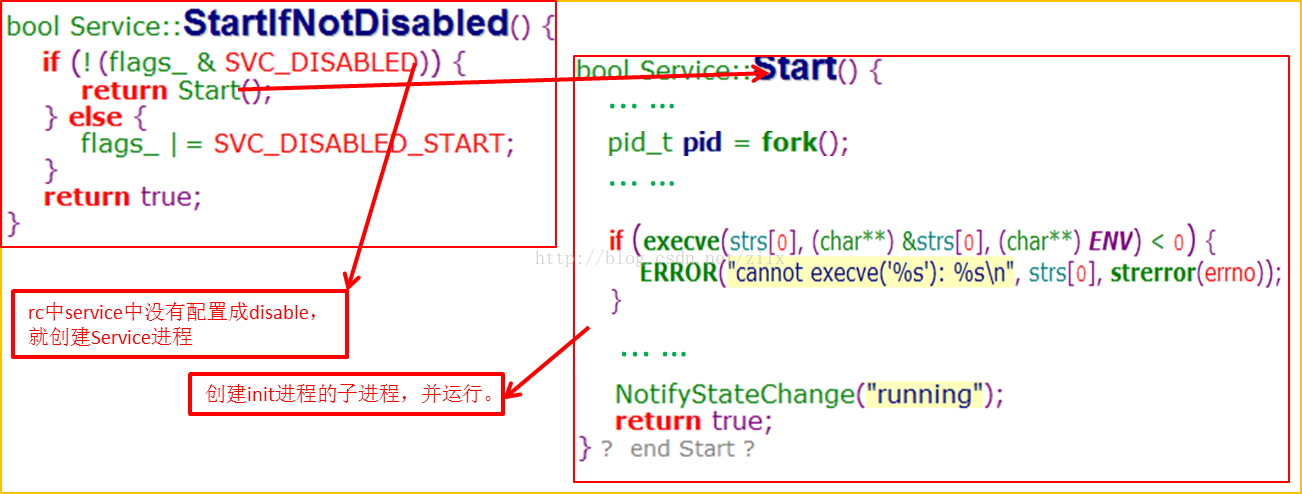


do\_class\_start在处理中会调用Service类的StartifNotDisabled()函数。





其中，Service-> StartifNotDisabled()函数调用Service->Start()函数，创建init进程的子进程，也即当前Service进程。



由此，zygote进程就被init进程创建并启动了。

通过上面介绍，我们大概知道了init中Service的启动过程。对于启动一个系统级Service，我们应了解：

1. Service的配置和启动控制都在Init.rc里。 只需对Init.rc进行配置就可以添加一个Service并启动。

2. Service section，生成Service对象并配置Service相关参数。

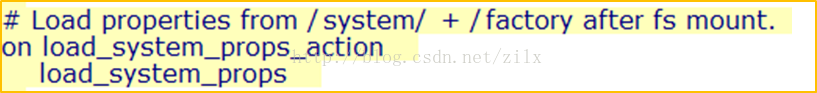
3. On section中的class\_start命令，用来启动Service。它的参数是需要启动的service 的class字段。

4. Service section中不能配置disable字段。

**System property的设置**

在init.rc和init进程中，我们注意到对system property也进行了配置并处理。

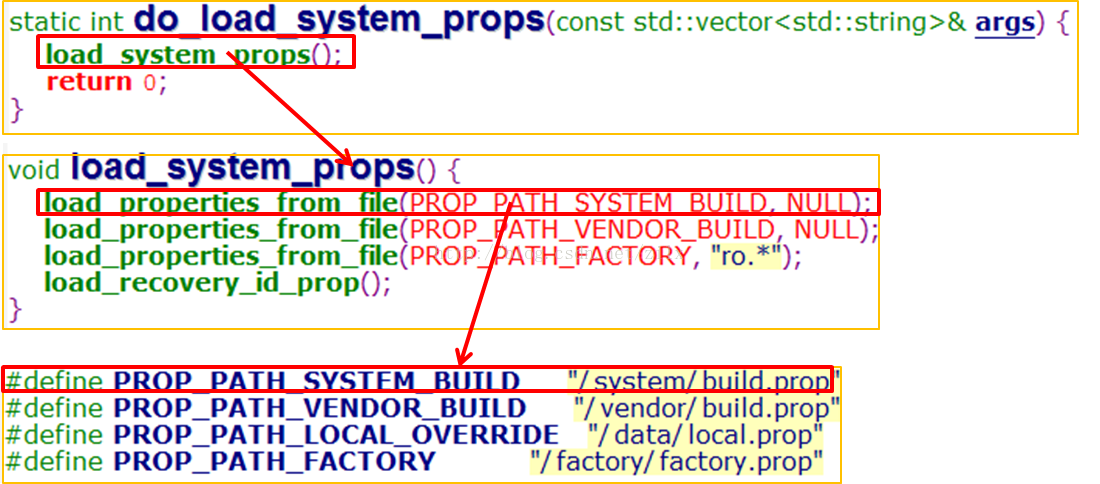
1.   Init.rc中配置action



2.  调用Builtins.cpp中的load\_system\_propsd执行函数

D:\360安全浏览器下载\20170514144609493.png

3.  执行do\_load\_system\_props（）函数，/system/build.prop等系统配置被加载并设置



**Init处理在Android开发中的注意点**

从init的处理得知，对于Android开发，我们可获知以下知识点以帮助应用程序的开发。

**1.  property及/system/build.prop**/system/build.prop在android开发中经常使用，它是在init进程中被处理。所以在程序开发的过程中，要充分考虑到这点。

**2.  用户添加系统级Service**

1）在init.rc中添加Service section的时候，尤其要注意class name，同时要注意on section中的class\_start命令，看看是否需要添加。因为他们直接关系到在开机启动后Service进程能否正常启动。

2）在zygote service配置中，有一个重要的选项：/system/bin/app\_process。该文件在/system/bin下，通过执行/system/bin/app\_process来进入app\_process(也就是zygote)进程处理。所以在添加Service配置的时候，一定要保证该类文件的存在。不然service是无法运行的。