**Android系统启动过程浅谈**

|  |  |  |
| --- | --- | --- |
| [日期：2011-10-11] | 来源：Linux社区  作者：yicao821 | [字体：[大](javascript:ContentSize(16)) [中](javascript:ContentSize(0)) [小](javascript:ContentSize(12))] |

**二、源码分析结果**

对[Android](http://www.linuxidc.com/topicnews.aspx?tid=11)-2.1版本中的如下源码文件进行分析：

(1) init.c：路径为system/core/init/init.c

(2) parser.c：路径为system/core/init/ parser.c

(3) builtins.c：路径为system/core/init/ builtins.c

(4) property\_service.c：路径为system/core/init/property\_service.c

(5) keycords.c：路径为system/core/init/keycords.c

(6) signal\_handler.c：路径为system/core/init/signal\_handler.c

总结得出系统启动流程对应的源代码文件及函数如下

**2.1 第一阶段(启动准备) system/core/init/init.c**

具体的函数执行过程如下：

|  |
| --- |
| mkdir -> mount -> open\_devnull\_stdio-> log\_init |

**2.2 第二阶段(解析init.rc文件)**

具体的函数调用过程如下：

|  |
| --- |
| init\_parse\_config\_file->parse\_config-> parse\_new\_section->parse\_service (或者parse\_action)-> parse\_line\_service(或者parce\_line\_action) |

**2.3 第三阶段(触发需要执行的action)**

具体的调用过程如下：

|  |
| --- |
| action\_for\_each\_trigger("early-init",action\_add\_queue\_tail);->action\_add\_queue\_tail ( class\_start default)->drain\_action\_queue();->action\_remove\_queue\_head -> do\_class\_start |

**2.4 第四阶段(执行在action队列中的命令)**

具体的调用过程如下：

|  |
| --- |
| **execute\_one\_command**-> action\_remove\_queue\_head->do\_class\_start->service\_for\_each\_class-> service\_start\_if\_not\_disabled-> service\_start |

**2.5 第五阶段(循环处理)**

具体的循环处理过程如下：

1. for (; ;) {
2. poll > handle\_property\_set\_fd > handle\_keychord > handle\_signal
3. }

**2.6 主要函数介绍**

|  |  |  |
| --- | --- | --- |
| 函数名 | 所在文件 | 功能概述 |
| main | system/core/init/init.c | 1号进程init的入口函数。主要分析init.rc配置文件，执行基本的action和启动必备的native service，然后进入一个infinite loop 处理来自property, signal的event |
| mkdir | system/core/init/init.c | 建立文件系统的基本目录 |
| mount | system/core/init/init.c | 装载文件系统 |
| open\_devnull\_stdio | system/core/init/init.c | 打开基本输入、输出设备 |
| log\_init | system/core/init/init.c | 初始化日志功能 |
| init\_parse\_config\_file | system/core/init/ parser.c | 读取init.rc文件内容到内存数据区 |
| parse\_config | system/core/init/ parser.c | 识别init.rc文件中的 Section（service and action series ）和Text |
| parse\_new\_section | system/core/init/ parser.c | 识别section类别 |
| parse\_service | system/core/init/ parser.c | 对service section第一行进行分析 |
| parse\_line\_service | system/core/init/ parser.c | 对service section的option选项进行分析 |
| parse\_action | system/core/init/ parser.c | 对action section第一行进行分析 |
| parse\_line\_action | system/core/init/ parser.c | 对action section的每一行独立的命令进行分析 |
| action\_for\_each\_trigger | system/core/init/ parser.c | 触发某个action的执行 |
| action\_add\_queue\_tail | system/core/init/ parser.c | 将某个action的从action\_list加到action\_queue |
| execute\_one\_command | system/core/init/init.c | 执行当前action的一个command |
| action\_remove\_queue\_head | system/core/init/ parser.c | 从action\_queue链表上移除头结点(action) |
| do\_class\_start | system/core/init/ builtins.c | class\_start default对应的入口函数，主要用于启动native service |
| service\_for\_each\_class | system/core/init/ parser.c | 遍历service\_list链表上的所有结点 |
| service\_start\_if\_not\_disabled | system/core/init/ builtins.c | 判断service的flag是否disabled，如果不是，则调用相关函数，准备启动service |
| service\_start | system/core/init/init.c | 启动service的主要入口函数，设置service数据结构的相关数据结构后，调用fork创建一个新的进行，然后调用execve执行新的service |
| fork | Lib function(ulibc) | 进程创建函数 |
| execve | Lib function(ulibc) | 调用执行新的service |
| poll | Lib function(ulibc) | 查询property\_set\_fd，signal\_fd和keychord\_fd文件句柄是否有服务请求 |
| handle\_property\_set\_fd | system/core/init/property\_service.c | 处理系统属性服务请求，如：service, wlan和dhcp等等 |
| handle\_keychord | system/core/init/keycords.c | 处理注册在service structure上的keychord，通常是启动service |
| handle\_signal | system/core/init/signal\_handler.c | 处理SIGCHLD signal |