**1．Init脚本语法**

fspad-733/androidL/system/core/rootdir/init.rc

打开这个文件后我们发现这个文件不是用c语言来写的。我们之前没有接触过这种语法，其实它叫init语法，具体的语法规则如下。

Android init脚本语言

前面简单分析了下init.c里的操作，里面提到了解析Init.rc和硬件脚本下面详细解析下Android init脚本语言的规范。

Android初始化语言包含了四种类型的声明：

    Actions（行为）、Commands（命令）、Services（服务）和Options（选项）

* 所有这些都是以行为单位的，各种记号由空格来隔开。
* C语言风格的反斜杠号可用于在记号间插入空格。
* 双引号也可用于防止字符串被空格分割成多个记号。
* 行末的反斜杠用于折行，注释行以井号（#）开头（允许以空格开头）。

Actions和Services声明一个新的分组Section。所有的命令或选项都属于最近声明的分组。位于第一个分组之前的命令或选项将会被忽略。

Actions和Services有唯一的名字。如果有重名的情况，第二个申明的将会被作为错误忽略。

**Actions（活动）**

  Actions其实就是一序列的Commands（命令）。Actions都有一个trigger（触发器），它被用于决定action的执行时间。当一个符合action触发条件的事件发生时，action会被加入到执行队列的末尾，除非它已经在队列里了。队列中的每一个action都被依次提取出，而这个action中的每个command（命令）都将被依次执行。

Actions的形式如下：

        on <trigger>

           <command1>

           <command2>

           <command3>

on后面跟着一个触发器，当trigger被触发时，command1，command2，command3，会依次执行，直到下一个Action或下一个Service。

简单来说，Actions就是Android在启动时定义的一个启动脚本，当条件满足时，会执行该脚本，脚本里都是一些命令commands，不同的脚本用on来区分。

Triggers（触发器）

    Triggers（触发器）是一个用于匹配特定事件类型的字符串，用于使Actions发生。

        boot：

            这是init执行后的第一个被触发的Triggers（触发器）。（在 /init.conf （启动配置文件）被装载之后）

        <name>=<value>：

            这种形式的Triggers（触发器）会在属性<name>被设置为指定的<value>时被触发。

        device-added-<path>：

        device-removed-<path>：

            这种形式的Triggers（触发器）会在一个设备节点文件被增删时触发。

        service-exited-<name>：

            这种形式的Triggers（触发器）会在一个特定的服务退出时触发。

触发器通常和on一起来联合使用。

示例：

on init

  export LD\_LIBRARY\_PATH /system/lib

  insmod modules/fsr.ko

  symlink /system/etc /etc

  mkdir /sdcard 0000 system system

  write /proc/cpu/alignment 4

on boot

  …

on property:ro.kernel.qemu=1

   start adbd

上面声明了三个action：init，boot和一个属性触发器，当init被触发时，会顺序执行后面的命令，直到on boot新的action。Init的触发是由init.c里的函数action\_for\_each\_trigger来决定的。当属性ro.kernel.qemu为1 时，会触发start adbd命令。

**Services（服务）**

Services（服务）是一个程序，它在初始化时启动，并在退出时可选择让其重启。Services（服务）的形式如下：

        service <name> <pathname> [ <argument> ]\*

           <option>

           <option>

           ...

name:服务名

pathname:当前服务对应的程序位置

option：当前服务设置的选项

**Options（选项）**

    Options（选项）是一个Services（服务）的修正者。他们影响Services（服务）在何时，并以何种方式运行。

     critical：

            说明这是一个对于设备关键的服务。如果他四分钟内退出大于四次，系统将会重启并进入recovery（恢复）模式。

     disabled：

            说明这个服务不会同与他同trigger（触发器）下的服务自动启动。他必须被明确的按名启动。

     setenv <name> <value> （设置环境变量）

            在进程启动时将环境变量<name>设置为<value>。

     socket <name> <type> <perm> [ <user> [ <group> ] ]

            创建一个Uinx域的名为/dev/socket/<name> 的套接字，并传递它的文件描述符给已启动的进程。<type> 必须是 "dgram"或"stream"。User 和 group默认为0。

     user <username>

            在启动这个服务前改变该服务的用户名。此时默认为root。（？？？有可能的话应该默认为nobody）。当前，如果你的进程要求Linux capabilities（能力），你无法使用这个命令。即使你是root，你也必须在程序中请求capabilities（能力）。然后降到你想要的 uid。

     group <groupname> [ <groupname> ]\*

            在启动这个服务前改变该服务的组名。除了（必需的）第一个组名，附加的组名通常被用于设置进程的补充组（通过setgroups()）。此时默认为root。（？？？有可能的话应该默认为nobody）。

     oneshot

            服务退出时不重启。

     class <name>

            指定一个服务类。所有同一类的服务可以同时启动和停止。如果不通过class选项指定一个类，则默认为"default"类服务。

     onrestart

            当服务重启，执行一个命令（下详）。

**Commands（命令）**

    exec <path> [ <argument> ]\*

         创建和执行一个程序（<path>）。在程序完全执行前，init将会阻塞。由于它不是内置命令，应尽量避免使用exec，它可能会引起init卡死。(??? 是否需要一个超时设置?)

    export <name> <value>

        在全局环境变量中设在环境变量 <name>为<value>。（这将会被所有在这命令之后运行的进程所继承）

    ifup <interface>

        启动网络接口<interface>

    import <filename>

           解析一个init配置文件，扩展当前配置。

    hostname <name>

           设置主机名。

    chmod <octal-mode> <path>

           更改文件访问权限。

    chown <owner> <group> <path>

           更改文件的所有者和组。

    class\_start <serviceclass>

           启动所有指定服务类下的未运行服务。

    class\_stop <serviceclass>

        停止指定服务类下的所有已运行的服务。

    domainname <name>

           设置域名。

    insmod <path>

           加载<path>中的模块。

    mkdir <path> [mode] [owner] [group]

           创建一个目录<path>，可以选择性地指定mode、owner以及group。如果没有指定，默认的权限为755，并属于root用户和root组。

    mount <type> <device> <dir> [ <mountoption> ]\*

        试图在目录<dir>挂载指定的设备。<device> 可以是以 [mtd@name](mailto:mtd@name) 的形式指定一个mtd块设备。<mountoption>包括 "ro"、"rw"、"remount"、"noatime"、 ...

    setprop <name> <value>

           设置系统属性 <name> 为 <value>值.

    setrlimit <resource> <cur> <max>

        设置<resource>的rlimit（资源限制）。

    start <service>

        启动指定服务（如果此服务还未运行）。

    stop <service>

        停止指定服务（如果此服务在运行中）。

    symlink <target> <path>

        创建一个指向<path>的软连接<target>。

    sysclktz <mins\_west\_of\_gmt>

        设置系统时钟基准（0代表时钟滴答以格林威治平均时（GMT）为准）

    trigger <event>

           触发一个事件。用于将一个action与另一个 action排列。

    write <path> <string> [ <string> ]\*

           打开路径为<path>的一个文件，并写入一个或多个字符串。

**Properties（属性）**

Init更新一些系统属性以提供对正在发生的事件的监控能力:

init.action

此属性值为正在被执行的action的名字，如果没有则为""。

init.command

此属性值为正在被执行的command的名字，如果没有则为""。

init.svc.<name>

名为<name>的service的状态("stopped"（停止）, "running"（运行）, "restarting"（重启）)