[android之init进程-uevent[转]](http://blog.itpub.net/7232789/viewspace-752133/) 2013-01-05 11:15:15

<http://blog.itpub.net/7232789/viewspace-752133/>

分类： Linux

Init是linux kernel启动的第一个进程，理解init,对熟悉android系统非常重要。

   Android的每个目录下面都有一个非常重要的文件Android.mk,负责编译该目录下面的代码。

   System/core/init/android.mk

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;"> <span style=< span="" style="word-wrap: break-word;">"font-size:18px;"><span style=< span="" style="word-wrap: break-word;">"color:#ff0000;">LOCAL\_MODULE:= init  </span style=<></span style=<></span style=<>
3. LOCAL\_FORCE\_STATIC\_EXECUTABLE := **true**
4. LOCAL\_MODULE\_PATH := $(TARGET\_ROOT\_OUT)
6. <span style="color:#ff0000;"</span style=>include $(BUILD\_EXECUTABLE)
8. SYMLINKS := $(TARGET\_ROOT\_OUT)/sbin/ueventd
9. $(SYMLINKS): INIT\_BINARY := $(LOCAL\_MODULE)
10. $(SYMLINKS): $(LOCAL\_INSTALLED\_MODULE) $(LOCAL\_PATH)/Android.mk
11. @echo "Symlink: $@ -> ../$(INIT\_BINARY)"
12. @mkdir -p $(dir $@)
13. @rm -rf $@
14. <span style="font-size:18px;color:#ff0000;">$(hide) ln -sf ../$(INIT\_BINARY) $@  </span style=

     上面的代码会生成一个叫init的可执行程序，它会被放在/下面，且同时    会产生一个符号链接/sbin/eventd,指向/init. 我们不禁要问，为什么这样做？

      Init是一个脚本解释器，它会对目标系统下的两个文件解析，

     /init.rc

     /init.xxx.rc (xxx代表平台平台名)

     先看看源代码目录/device/xxx/init.rc

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;">  on early-init  </span style=<>
2. start ueventd

     看来init在解析脚本的时候又启动了一个自己的进程，只是进程名变成了ueventd.

     System/core/init/init.c/main

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;">  **if** (!strcmp(basename(argv[0]), "ueventd"))  </span style=<>
2. **return** ueventd\_main(argc, argv);

     根据进程名不同，程序执行路径不同。Ueventd顾名思义应该是接收 uvent的守护进程,这里它的主要作用根据uevent是创建或删除/dev/xxx(xxx设备名)，我们知道在linux下面创建设备节点的接口 mknod,我们跟进去看看这个接口是在哪里调用的

     System/core/init/Ueventd.c/ueventd\_main

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;">ueventd\_parse\_config\_file("/ueventd.rc");  </span style=<>
3. snprintf(tmp, **sizeof**(tmp), "/ueventd.%s.rc", hardware);
4. ueventd\_parse\_config\_file(tmp);
6. device\_init();

          ueventd有两个脚本需要解析,ueventd.rc,ueventd.xxx.rc,脚本，又见脚本这个脚本可以让客户设置/dev 或 /sys目录及子目录的权限.

    system/core/rootdir/ueventd.rc

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;">  /dev/binder               0666   root       root  </span style=<>

     这里请注意，ueventd\_parse\_config\_file并不创建设备节点，它的作用是提供数据库，当有设备节点生成的时候，eventd会参考这个数据库设置设备节点的权限。

      system/core/init/devices.c/device\_init

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;"> device\_fd = open\_uevent\_socket();  </span style=<>
2. coldboot("/sys/class");
3. coldboot("/sys/block");
4. coldboot("/sys/devices");

       这个函数很简单，主要是创建了uevent的socket handle,同时触发/sys/clas,/sys/block,/sys/devices这三个目录及其子目录下的uevent，然后接受并创建设备节点，至此设备节点才算创建，coldboot里面有个很有意思的函数do\_coldboot，这是一个递归调用函数，实现的很有意思，大家可以看看.

     system/core/init/Ueventd.c/ueventd\_main

**[cpp]** [view plaincopy](http://blog.csdn.net/harbordaisy/article/details/7198494)

1. <span style=< span="" style="word-wrap: break-word;">"font-size:18px;">  **while**(1) {  </span style=<>
2. ufd.revents = 0;
3. nr = poll(&ufd, 1, -1);
4. **if** (nr <= 0)
5. **continue**;
6. **if** (ufd.revents == POLLIN)
7. handle\_device\_fd();
8. }

          死循环，接受kernel传过来的uevent，动态创建或删除节点.

      handle\_device\_fd会最终调用mknod创建设备节点，流程如下:

      handle\_device\_fd-> handle\_device\_event-> make\_device-> mknod

      over.