[转]Android Recovery 工作流程

09年初写的Android Recovery

By codingguy on 2011 年 12 月 14 日

找到了以前写的老文档，09年初写的，不管内容怎样，贴出来晒晒~

————————————————————————

Android Recovery

Android Recovery: 功能简介

Android支持Recovery模式。在某些操作之后，系统会自动重启并进入到Recovery模式，用户按组合键开机（HOME+POWER），也可进入Recovery模式。该模式提供如下功能：

1、擦除用户数据

恢复系统到出厂模式，即擦除用户数据和缓存数据。

2、系统升级

系统升级的概念比较广，包括系统文件的升级、恢复损害的系统数据、firmware的升级，以及应用软件的维护，甚至影音文件的下载。系统升级需要使用特定的升级包，Android使用OTA[1]升级包，其初衷在于可以发挥广域无线通信链路的优势，如3G。

升级方式有两种：

1、在线升级
利用无线通信网络，系统自动连接更新源，查看有无升级包、下载OTA升级包，然后给出提示，发起升级过程，如下左图。感觉有点类似Windows XP的系统更新，只不过升级的时候，Android系统会重启系统进入Recovery模式。另外Android的升级内容很广泛，比如可以通过这种方式安装应用程序。T-Mobile已经提供了这种服务，如升级服务器以OTA无线方式向G1终端发送Android平台RC33升级包，传输媒介可以是3G网络、Wi-Fi或GPRS。

2、离线升级

可以将下载到的OTA包放在SD卡里，通过离线方式升级，如下右图所示。这种升级方式比较灵活，不用花费无线流量。这样一来，使用自己制作的OTA进行升级也成为可能。事实上，G1就是用这种方式进行刷机的，比如更新radio firmware以支持某个频段。

Android: 分区结构

在分析Recovery工作流程之前，我们先了解一下Android文件系统的分区结构。下表是android/bootable/recovery/root.c中提得到的结构：

Name	Device	Partition	Mount	File system

pmp 旅游攻略

最新文章

『金丹大道』玉皇心印妙...
回风混合 百日功灵
UART硬件流控制信号的使用（图）
認識Android環境裡的兩種Servic...
国家标准经穴部位挂图*
[z]Cordova(PhoneGap)体系结构(...
更多

热门文章

中国人！我们严重误判了日本民...
令人叫绝的家传秘方
开网店流程（2014年）
毛泽东怎样写文章
超神奇简单腿功——锻炼数日后...
一个躺在床上的老男人对儿子说...
好想上去问姑娘，这是不是今年...
《一个女人成功嫁人经验》每个...

惬意大美女
天天饮食菜谱合集
PPT要这样做才漂亮，专业级！
去肝火的饮食调养法 几种清火茶...
更多>>

		name	point	
BOOT	g_mtd_device	Boot	NULL	g_raw
CACHE	g_mtd_device	Cache	/cache	yaffs2
DATA	g_mtd_device	Userdata	/data	yaffs2
MISC	g_mtd_device	Misc	NULL	g_raw
PACKAGE	NULL	NULL	NULL	g_package_file
RECOVERY	g_mtd_device	Recovery	/	g_raw
SDCARD	/dev/block/mmcblk0p1	NULL	/sdcard	Vfat
SYSTEM	g_mtd_device	System	/system	yaffs2
TMP	NULL	NULL	/tmp	NULL

Root file system layout

模拟器环境下adb shell里的mount输出：

```
# mount

……

/dev/block/mtdblock0    /system           yaffs2    ro                 0    0

/dev/block/mtdblock1    /data             yaffs2    rw,nosuid,nodev    0    0

/dev/block/mtdblock2    /cache            yaffs2    rw,nosuid,nodev    0    0
```

综上，MTD中有如下分区：

BOOT： boot.img，Linux kernel (within normal ramdisk)

MISC： bootloader message struct

RECOVERY： recovery.img，Linux kernel (within recovery ramdisk)

SYSTEM： system.img

DATA： userdata.img

CACHE： some cache files

有几点说明：

1、一般来讲，主板上还有用于存储bootloader的可擦写存储设备。若具备通信能力，还要存储radio firmware，这两部分的更新由Recovery协助Bootloader完成，没有代码证明一定存在NAND flash上。

2、RECOVERY分区无文件系统，存放二进制image。

3、SYSTEM中有recovery.img的备份：/system/recovery.img，initrc中有如下代码：

```
service flash_recovery /system/bin/flash_image recovery system/recovery.img

oneshot
```

每次启动，flash_image程序，会检查recovery分区中image的header，如果与备份的recovery.img不符，就会把备份写到RECOVERY分区。这样做是为了应对RECOVERY分区遭到破坏。当然，我们也可以更换这个备份，这样也会将其写到RECOVERY。事实上，处于安全及版权考虑，OTA是有签名的（其实就是JAR包），Recovery对签名有要求，所以只能进行被允许的升级，此时的破解思路就是更换一个不检查签名的Recovery程序，方法就是设法更换/system/recovery.img。

Android Recovery: 三个部分、两个接口

Recovery的工作需要整个软件平台的配合，从架构角度看，有三个部分：

1、Main system：用boot.img启动的Linux系统，Android的正常工作模式。

2、Recovery：用recovery.img启动的Linux系统，主要是运行Recovery程序。

3、Bootloader：除了加载、启动系统，还会通过读取flash的MISC分区获得来自Main system和Recovery的消息，并以此决定做何种操作。

在Recovery的工作流程中，上述三个实体的通信必不可少。通信的接口有以下两个：

l CACHE分区中的三个文件：/cache/recovery/…

Recovery通过/cache/recovery里的文件与main system通信，有三个文件：

1. /cache/recovery/command

Main system传给Recovery的命令行，每一行有一个命令，支持以下几种：

–send_intent=anystring write the text out to recovery/intent

–update_package=root:path verify install an OTA package file

–wipe_data erase user data (and cache), then reboot

–wipe_cache wipe cache (but not user data), then reboot

1. /cache/recovery/log

Recovery的log输出，在recovery运行过程中，stdout及stderr会重定位到/tmp/recovery.log文件，Recovery退出之前会将其转储到/cache/recovery/log中，也就是cache分区的recovery/log。

1. /cache/recovery/intent

Recovery传给Main system的信息

l BCB (bootloader control block)

struct bootloader_message {

char command[32];

char status[32];

char recovery[1024];

};

BCB是Bootloader与Recovery的通信接口，也是Bootloader与Main system的通信接口，存储在flash中的MISC分区，占用三个page，各成员意义如下：

command：

当想要重启进入recovery模式，或升级radio/bootloader firmware时，会更新这个域。当firmware更新完毕，为了启动后进入recovery做最终的清除，bootloader还会修改它。

status：

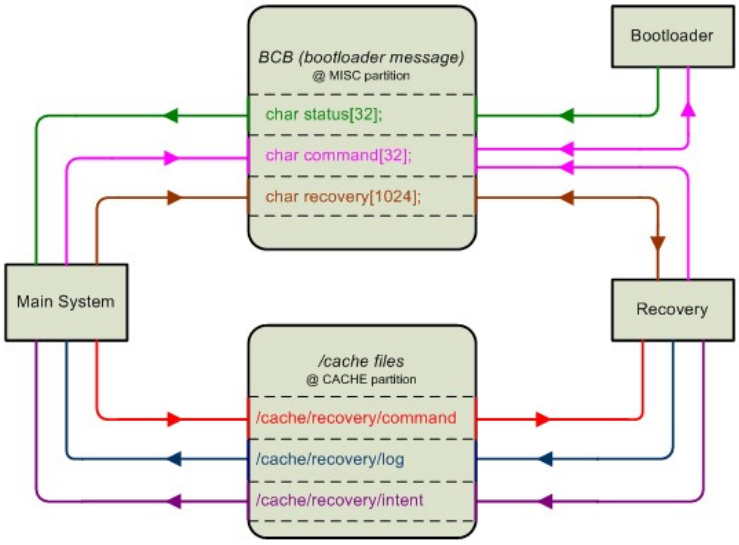
update-radio或update-hboot完成后，bootloader会写入相应的信息，一般是一些状态或执行结果。

recovery：

recovery：

仅被Main system写入，用于向Recovery发送消息，必须以“recovery\n”开头，否则这个域的所有内容会被忽略。这一项的内容中“recovery/\n”以后的部分，是/cache/recovery/command支持的命令，可以认为这是在Recovery操作过程中，对命令操作的备份。Recovery也会更新这个域的信息，执行某操作前把该操作命令写到recovery域，并更新command域，操作完成后再清空recovery域及command域，这样在进入Main system之前，就能确保操作被执行。

如图所示，Main system、Recovery与Bootloader通过上述接口通信，通信逻辑依不同的目的而不同，在后面介绍具体工作流程中还会详细介绍。



从Main system进入Recovery的方法

我们提到，从Main system进入到Recovery，要修改MISC分区的数据并重启，从而告诉Bootloader是用boot.img还是用recovery.img启动。

init.c里的wait_for_one_process函数中有如下代码：

__reboot(LINUX_REBOOT_MAGIC1, LINUX_REBOOT_MAGIC2,

LINUX_REBOOT_CMD_RESTART2, “recovery”);

一些关键的进程运行异常，会重启进入recovery模式，这里用__reboot函数进入recovery。跟

踪这个函数，由系统调用处理函数，到kernel_restart(char *cmd)，最终调用machine_restart使用体系结构相关的代码完成重启。

Android中没有给出如何处理“recovery”重启。不过可以断定，在重启之前会向BCB中写入信息，以告知bootloader如何启动，具体操作是这样的：

向command域中写入“boot-recovery”　// 此操作必做

向recovery域写入“recovery\n”　// 此操作也可不做

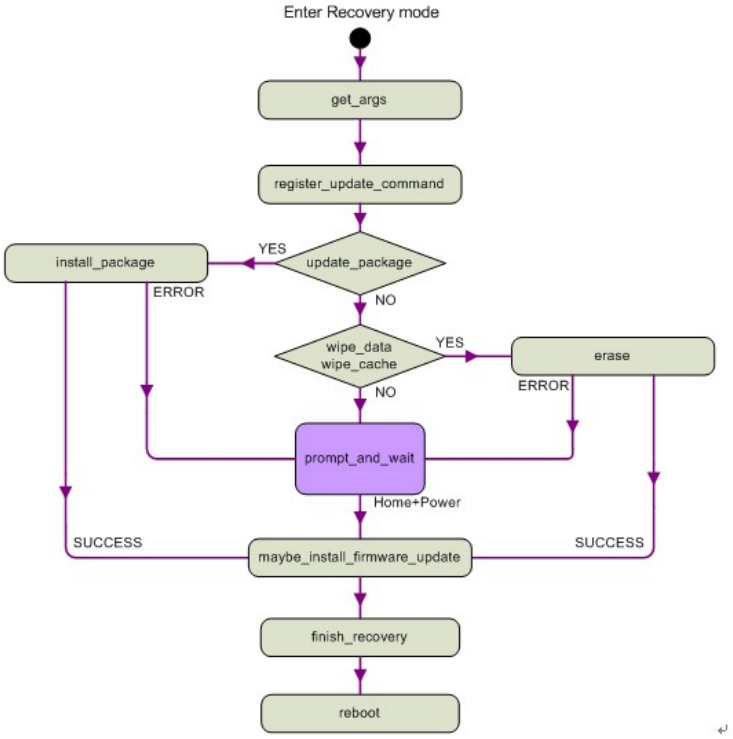
这些操作很可能在kernel_restart(char *cmd)中完成，因为这一部分与体系结构无关，如果要实现完整的Recovery，这部分工作是必须做的。

Bootloader得到进入Recovery模式的指示，用recovery.img启动，进入Recovery模式，init.rc (bootable/recovery/etc/init.rc)的内容比Main system的要短的多，最重要的是把recovery程序作为服务启动：

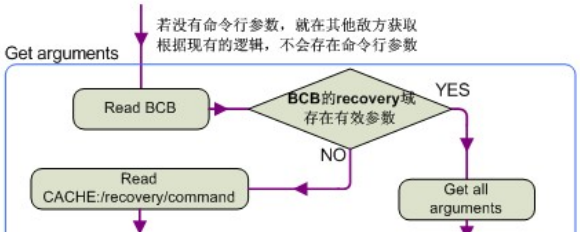
service recovery /sbin/recovery

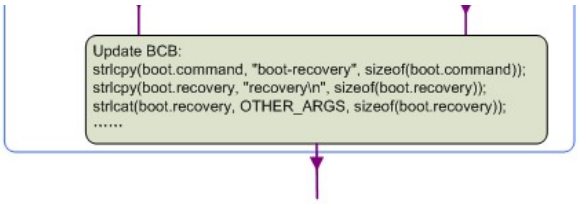
Android Recovery: 总体流程

根据Recovery的initrc，kernel启动完成后，启动recovery服务，这是一个C程序，入口在/bootable/recovery/recovery.c中，main函数结构清晰，主要流程如图：



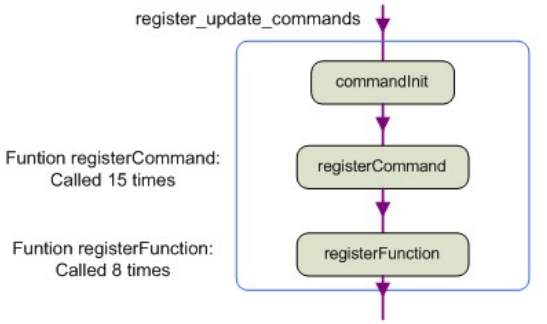
l　get_args：首先调用get_args获取参数，主要流程如下：





get_args不仅传回获取到的参数，还会将其写入BCB，这样，一旦升级或擦除数据的过程中出现错误，重启之后依然进入Recovery并做相同操作。

l register_update_command，这是为update做准备工作，负责注册update用的command & function，正是这些command & function组成了update用到的update_script：



先用commandInit（android/bootable/recovery/amend/command.c）初始化command symbol table，然后多次调用registerCommand及registerFunction注册command及function。command相关的源代码都在amend目录中，语法的构建及解析使用Android已经包含的Bison（Yacc）。

这里的command有15个，见下表：

Command Name	Argument Type	Command Handler
assert	CMD_ARGS_BOOLEAN	cmd_assert
delete	CMD_ARGS_WORDS	cmd_delete
delete_recursive	CMD_ARGS_WORDS	cmd_delete
copy_dir	CMD_ARGS_WORDS	cmd_copy_dir
run_program	CMD_ARGS_WORDS	cmd_run_program
set_perm	CMD_ARGS_WORDS	cmd_set_perm
set_perm_recursive	CMD_ARGS_WORDS	cmd_set_perm
show_progress	CMD_ARGS_WORDS	cmd_show_progress
symlink	CMD_ARGS_WORDS	cmd_symlink
format	CMD_ARGS_WORDS	cmd_format
write_radio_image	CMD_ARGS_WORDS	cmd_write_firmware_image
write_hboot_image	CMD_ARGS_WORDS	cmd_write_firmware_image
write_raw_image	CMD_ARGS_WORDS	cmd_write_raw_image
mark	CMD_ARGS_WORDS	cmd_mark
done	CMD_ARGS_WORDS	cmd_done

CMD_ARGS_BOOLEAN表示该command后面接的参数是boolean值，即true或false，解析脚本时计算参数的逻辑值，然后传给command handler，目前只有“assert”这个command用此类型的参数。

CMD_ARGS_WORDS表示该command后面接的参数是字符，形如C程序启动时加的参数，解析脚本时把参数直接传递给command handler，比如“format BOOT:”，“BOOT:”会传给

cmd_format。

Function Name	Function Handler
compatible_with	fn_compatible_with
update_forced	fn_update_forced
get_mark	fn_get_mark
hash_dir	fn_hash_dir
matches	fn_matches
concat	fn_concat
getprop	fn_getprop
file_contains	fn_file_contains

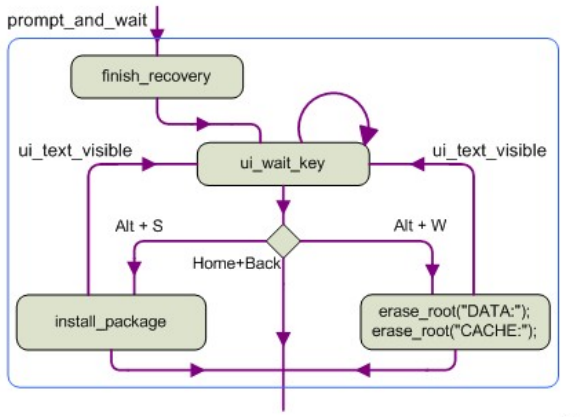
function与command用同样的处理框架，只不过function会产生返回值，目前见到的用法一般都

是与assert一起使用，例如下面脚本：

assert getprop(“ro.bootloader”) == “0.95.0000″

先用getprop从properties中取得bootloader版本，然后再将比较后的boolean值传给assert。

l　　prompt_and_wait：等待用户输入



首先打印文本信息。然后执行finish_recovery(NULL)，这个函数后面介绍。然后进入ui_wait_key等待用户输入，按下不同的组合键会有不同的动作。对于键盘输入，先到达input_thread函数(android/bootable/recovery/ui.c)，在那里处理两种组合键，其余才交给ui_wait_key处理：

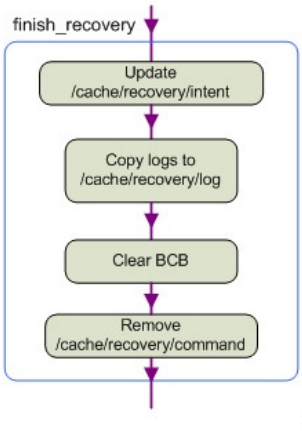
KEY	Funcion	Handler
Home + Back	reboot system now	ui_wait_key
Alt + S	apply sdcard:update.zip	ui_wait_key
Alt + W	wipe data/factory reset	ui_wait_key
Alt + L	toggle log text display	input_thread
Green + Menu + Red	reboot immediately	input_thread

Home + Back：退出prompt_and_wait。

Alt + W或Alt + S，执行完install_package或erase_root后，若没有激活log text display，那么，就会退出prompt_and_wait，否则继续等待输入。

Green + Menu + Red：立刻重启，一般这样还会进入Recovery，因为BCB还没有来得及清空。

l finish_recovery：离开Recovery进入Main system的必经之路，流程如下：



intent内容作为参数传进来，如果有intent需要告知Main system，将其写入/cache/recovery/intent；

将所有log信息转储到/cache/recovery/log文件，以供Main system读取；

清除BCB，也就是告知Bootloader启动进入Main system；

删除/cache/recovery/command；

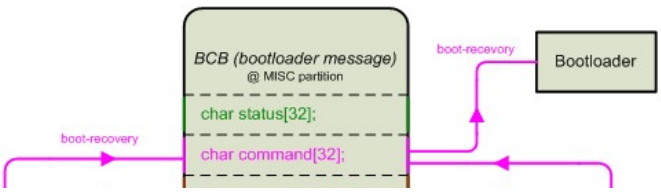
以上是整体流程中的几个函数，关于安装升级包、升级firmware等操作将在具体流程中介绍。

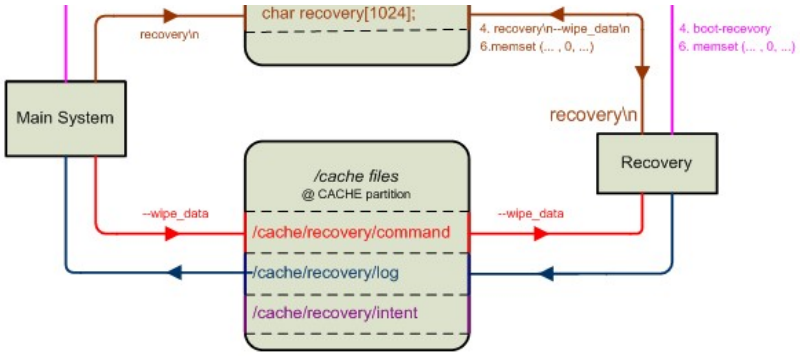
Android Recovery: Factory data reset流程

如果系统不稳定，可以尝试恢复出厂设置，该操作会擦除DATA分区及CACHE分区，有两种恢复方式，下面分别介绍：

l 通过Setting程序发起Facory data reset：

屏幕显示如上图，结合着下面的通信图，列出工作流程：





1、在应用程序Setting中选择factory data reset

2、Main system向/cache/recovery/command写入”–wipe_data”

3、Main system重启进入recovery模式（方法：修改BCB）

4、Recovery向BCB写入”boot-recevory”和”recovery\n–wipe_data\n”

5、擦除DATA分区，里面是用户数据，擦除CACHE分区

6、finish_recovery函数

7、重启，回到Main system

第四个步骤，Recovery向BCB写入boot-recovery和–wipe_data，这是为了保证后面几个步骤的完整执行。如，在擦除DATA分区或CACHE分区过程中，如果发生了重启、关机等操作，导致没有擦除成功，那么再次用常规方式开机后，Bootloader会依据BCB的指示，引导进入Recovery，并重新擦除这两个分区。擦完DATA分区与CACHE分区后，调用finish_recovery，做返回Main system前最后的工作，最终要的是擦除BCB，即MISC分区。此后，用常规方式重新开机，系统会进入Main system。

阅读Android的代码，发现Setting通过RPC调用Checkin Service的masterClear()启动这个过程，然而在Android中并没有找到masterClear()的实现，相关代码需要在产品化的过程中加入。从ICheckinService.aidl的注释可以了解到这个函数的作用：

/** Reboot into the recovery system and wipe all user data. */

代码位置：

/packages/apps/Settings/src/com/android/settings/MasterClear.java

/frameworks/base/core/java/android/os/ICheckinService.aidl

l 通过HOME+POWER组合键进入Recovery，再按ALT+W启动Factory data reset

过程比较简单，而且与上一种方式类似，结合总体流程，步骤如下：

1、捕获按键Alt + W。

2、擦除DATA分区、擦除CACHE分区。

3a、若激活了log显示（ALT+L：toggle log text display），调用finish_recovery函数重启，回到Main system。

3b、若没有激活log显示，继续接收按键，可用HOME+BACK重启回到Main system。

Android Recovery Update: 流程

l update.zip

update操作需要升级包，该升级包是文件名是*.zip，但观察包内结构会发现其实就是JAR包，JAR包是具有特定目录和文件结构的ZIP压缩包，因此可以作为ZIP包解开：

MANIFEST.MF：这个manifest文件定义了与包相关数据。

XXX.SF：这是JAR文件的签名文件，占位符xxx标识签名者，如CERT。

XXX.DSA：与签名文件相关联的签名程序块文件，它存储了用于签名JAR文件的公共签名。

在META-INF/com/google/android目录下有update_script文件，内容就是update要做的操作，也就是前面提到过的command序列。

出于安全性及版本控制的考虑，JAR包要求必须有完整性以及合法性签名。可以看出这是Android确保安全的策略。JAR相关内容参见http://www.ibm.com/developerworks/cn/java/j-jar/，这里就不再详细介绍。

l Main system部分

通过Android系统下载升级包并启动升级操作，需要上层应用Updater的支持，它是Java程序，代码位置android/packages/apps/Updater。大致流程：

系统启动后，如果存在网络连接，则检查是否存在升级包；

如果存在升级包，则下载至/cache目录；

调用Updater程序来提示是否升级；

如果Updater程序进程不存在,则自动启动此程序；

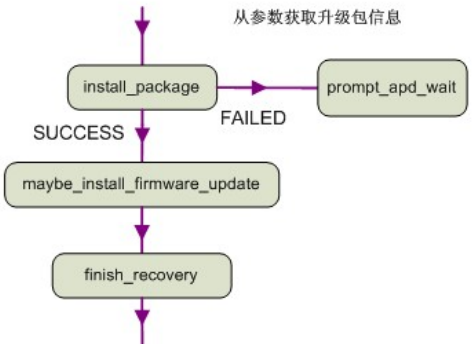
没有在代码中找到开始升级后执行哪些操作。不过由recovery.c的注释部分可以肯定一定需要重启进入Recovery，重启前要更新/cache/recovery/command，以告知Recovery进行升级：

–update_package=root:path

l update流程

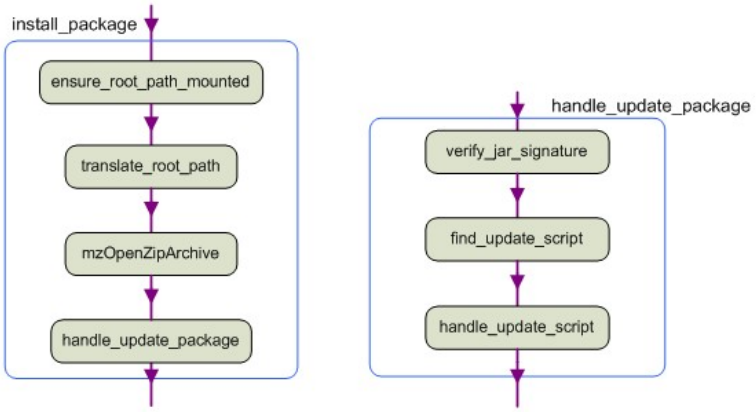
update有两种方式，第一种是上面提到的由Android启动的自动update过程，升级包在cache/

下，升级包的名字在/cache/recovery/command文件中指定。第二种是手动进入Recovery模式，然后输入Alt + S，安装/sdcard/update.zip升级包。两种方式不同的只是安装包的位置以及传递参数给Recovery的方法，update过程都是一样的，工作流程如下图所示：



install_package @ android/bootable/recovery/install.c

得到安装包信息，如“–update_package=CACHE:update.zip”，进入install_package函数，流程如下左图。mount安装包所在的分区，然后打开zip压缩包，进入handle_update_package开始升级：



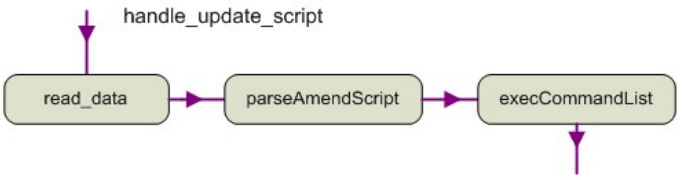
handle_update_package中，先对包进行校验，校验过程分三步：

verifySignature： 检验SF文件与RSA文件的匹配

verifyManifest： 检验/META-INF/MANIFEST.MF与签名文件中的digest是否一致

verifyArchive： 检验包中的文件与MANIFEST是否一致

接着find_update_script从MANIFEST.MF找到update_script的位置，然后handle_update_script，如下图，把内容读到buffer后，对其进行解析，分解成各个command（包括function）放在一个list中依次执行。

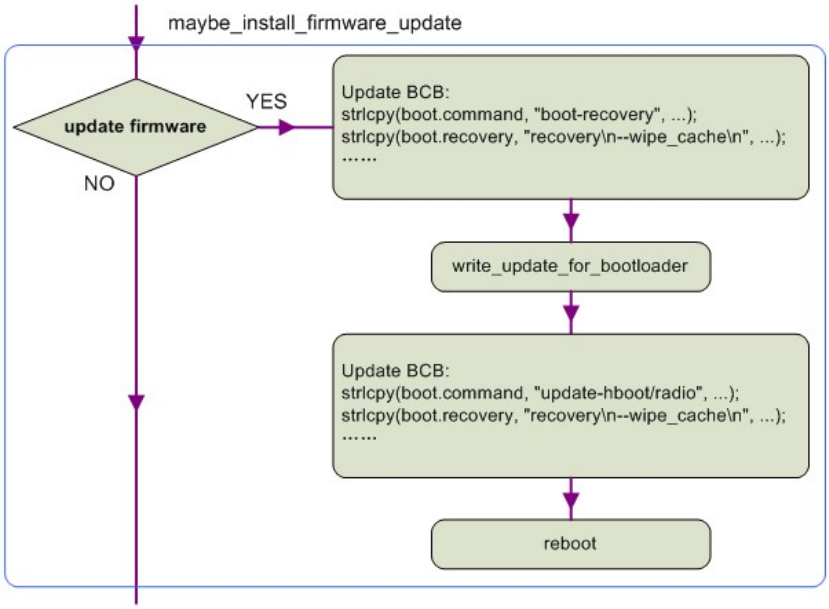


maybe_install_firmware_update @ android/bootable/recovery/firmware.c

install_package成功后，调用maybe_install_firmware_update，这个函数处理firmware的更新。update firmware脚本是这样的：

write_radio_image PACKAGE:radio.img

cmd_write_firmware_image处理write_radio_image这个命令，将image从压缩包加载到RAM中，并调用remember_firmware_update更新update_type、update_data及update_length。这三个变量对于maybe_install_firmware_update是可见的，并由它们来判断是否要安装firmware。下面是主要流程：



如果升级涉及radio / hboot firmware （radio：基带处理相关，hboot：bootloader）

1、向BCB写入”boot-recovery”和”–wipe_cache”

……此后重启系统，将进入recovery并擦除CACHE分区

2、write_update_for_bootloader向raw CACHE分区写入image，CACHE分区的内容将被破坏。

3、向BCB写入”update-radio/hboot”和”–wipe_cache”

4、重启，由Bootloader更新firmware

5、Bootloader向BCB写入”boot-recovery”，并保留BCB中recovery里的”–wipe_cache”

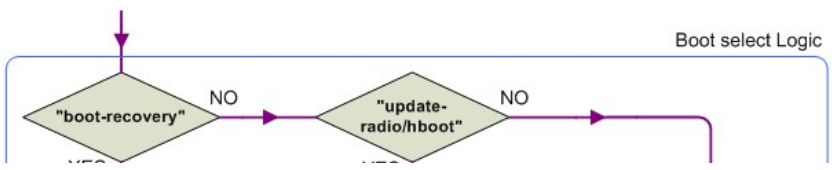
6、重启，再次进入Recovery，调用erase_root()擦除CACHE分区

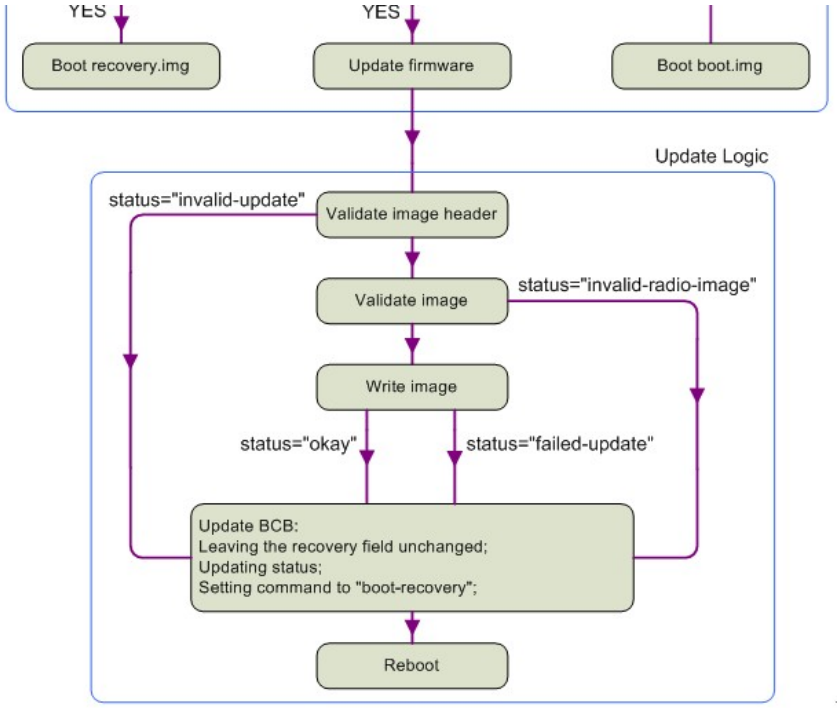
7、finish_recovery()清除BCB

8、重启，进入main system

l Bootloader

每次启动，Bootloader都会读取位于MISC分区的bootloader_message，并检查command区域以\0结尾，还要考虑flash存在坏块的情况。然后根据读取的命令，启动系统或者更新firmware。工作流程如下：





升级之后，无论升级成功是否，Bootloader都会进入recovery完成最后的收尾工作，并带着status以告知是否成功。如果更新hboot（尚不知道为什么叫这个名字，不过可以确定它就是bootloader firmware），一旦失败，若原有的bootloader遭到破坏，那么系统将不能boot。

为实现Android Recovery，还需要做什么？

实现Setting中Factory data reset

查看Updater工作流程，找到发起update的方法

实现__reboot(…..”recovery”)函数，连接Main system与Recovery

升级包的打包方法，以及JAR包签名机制

实现Bootloader与Recovery及Main System的通信；

实现Bootloader的启动逻辑、firmware升级；

---

[1] OTA：Over The Air，一种手机等终端应用的“空中下载”技术，利用这种技术用户可以通过下载来修补终端的漏洞或升级某些功能

来自：techres > 《androidKernel&Driver》
上一篇：[z]kthreadd khelper 内核线程
下一篇：mx53

Android recovery 系统

Android系统基础普及：Bootloader、Reco...

Android系统Recovery工作原理之使用upda...

Android刷机原理之recovery内部工作原理...

android recovery模式及ROM制作

Qualcomm Android recovery

共 1 条评论

linuxdog 01-14 19:36

0 0

太牛了！

回复

发表评论：

您好，请 **登录** 或者 **注册** 后再进行评论

其它帐号登录：