Android recovery 流程分析

一 Recovery 模式主要提供两种功能:

1 恢复出厂。上层发 ACTION_MASTER_CLEAR 的广播，MasterClearReceiver.java 中接收广播，最终调用 RecoverySystem.rebootWipeUserData()方法。这个方法会往 /cache/recovery/command 中写—wipe_data 命令。

2 系统升级。包括本地升级，OTA 升级和 fireware 升级。

二 进入 recovery 模式两种方式:

1 按键，通常是组合键或者音量加减键，按着开机，不同设备进入条件不一样。以某种按键的方式进入 recovery 的本质都是往 BCB 里面写命令

2 通过上层写命令触发

通过上面的判断，kernel 会选择加载 recovery.img 还是 boot.img。进入 recovery 模式后装载 recovery 分区，recovery.img 包含标准内核和根文件系统,执行的第一个进程就是 init，该进程会读取 init.rc 启动相应的服务。并执行 recovery 服务(/sbin/recovery)来做对应的操作。

三 源码分析

recovery 源码目录 bootable/recovery。

编译后文件生成目录 out/target/product/rk3368/recovery/root

1 重定向 log 到/tem/recovery.log，redirect_stdio();

如果有单一参数—adbd，执行 adb sideload 分支

2 加载分区表，load_volume_table()

fs_mgr_read_fstab() 读取分区表/etc/recovery.emmc.fstab

fs_mgr_add_entry()对应信息填充 fstab 结构体

打印信息：编号 ，挂载节点，文件系统类型，块设备长度

fs_mgr 模块源码位于/system/core

3 获取内外 sd 卡路径，SetSdcardRootPath()

通过 property_get 获取系统属性获得

4 打印参数，dumpCmdArgs(),从 sd 卡或者 usb 获取参数

`get_args()`不仅传回获取到的参数，还会将其写入 `BCB`，一旦升级或擦除数据过程中出错，重启后依然进入 `recovery` 并做相同操作。

5 参数解析，getopt_log(),命令行解析函数，解析传进来的命令，并对相应的变量赋值。

6 显示菜单，show_text()

recovery 界面代码 device/rockchip/common/recovery/recovery_ui.cpp

7 设置 selinux 权限

8 StartRecovery()不知道干嘛的

9 安装包，install_package()

把文件流读到 tmp 目录下的一个文件中去

load_keys()不知道干嘛的
verify_file()从升级包中查找签名文件看是否和给的 key 相匹配
mzOpenZipArchive()解压 zip 文件
try_update_binary()

10 是否擦 data 和 cache,如果擦 cache 会先把/cache/recovery/里面以前和现在的 log 加载到内存，格式化后再写入 cache，最终格式化调用的是 format_volume()，实际是用 ioctl 完成的擦除操作

11 prompt_and_wait()

如果升级失败，会出现 UI 页面，等待用户输入

12 finish_recovery()

清除 BCB 升级命令，并重启，拷贝升级 log

四 重点分析两个函数

1 分析 get_args()

get_bootloader_message()从 misc 分区读命令，如果读到了填充到参数里，如果读不到再从/cache/recovery/command 里面读，最后用 set_bootloader_message()将参数信息写回 BCB.

2 分析 try_update_binary()

两个关键的文件：update-script 和 update-binary
update-script 记载着系统升级所需要执行的命令
而 update-binary 则是对于每条命令的解析
update-script 是 update-binary 文件的文本形式
while 循环只负责更新
bootable/recovery/install.c 中定义了对应于每条命令的执行函数
通过调用 fork 函数创建出一个子进程，在子进程中开始读取并执行升级脚本文件 `execv(binary, args);`。在此需要注意的是函数 fork 的用法，fork 被调用一次，将做两次返回，在父进程中返回的是子进程的进程 ID，为正数；而在子进程中，则返回 0。所以 execv 实际是子进程中所进行的操作。子进程创建成功后，开始执行升级代码，并通过管道与父进程交互，将 pipefd[1]作为参数传递给子进程，子进程则将相关信息写入到此管道描述符中；而父进程则通过读取子进程传递过来的信息更新 UI。

五 通信

1 主系统与 recovery 的通信是通过/cache 下三个文件：

a /cache/recovery/command：保存 main system 传给 recovery 的命令行
b /cache/recovery/log：保存 recovery 模式下的 log 信息
c /cache/recovery/intent：recovery 传给 main system 的信息，作用不知道

上层可能过调用 android 标准的 RecoverySystem 类(frameworks/base/core/java/android/os/RecoverySystem.java)的接口来完成与 recovery 的通信。可执行安装(installPackage)，重置(rebootWipeUserData)，清缓存(rebootWipeCache)等操作。

2 bootloader 与 recovery 通过 BCB(Bootloader Control Block)通信

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 的 misc 分区，占用三个 page，本身是一个结构体，位于/bootloader/recovery/bootloader.h 文件中：

```
struct bootloader_message {
    char command[32];
    char status[32];
    char recovery[768];
    char stage[32];
    char reserved[224];
};
```

recovery 内容以 recovery\n 开头，一行一条命令。