升级方式：SD卡、usb、OTA升级

（以OTA升级方式梳理整个升级流程）

在setting中触发升级按钮，检测prop中的服务器路径上的版本是否比本地版本高，如果低于或者等于本地版本，则提示已是最新版本，如果高于本地版本，则提示有新版本是否进行升级，点击立即升级。

bootCommand（）此函数在main system重启之前做准备工作。首先创建/cache/recovery/目录，删除该目录下的command和log（可能不存在）在sqlite数据库中的备份，然后将上一步的arg命令写入到/cache/recovery/command文件中，下一步就真正重启了。

源码路径：frameworks/base/core/java/android/os/RecoverySystem.java

BCB：bootloader control block

arg就是进入recovery模式后，recovery服务需要进行的操作

RecoverySystem类中的**installPackage(Context context,FilepackageFile)**函数根据传进来的包文件获取包的绝对路径filename，然后将其拼成**arg=“--update\_package=”+filename**，它最终会被传到BCB中，然后被传到**bootCommand(context,arg)。**

升级包在此处也被传进来了

mService是SystemUpdateService实例，跟踪发现SystemUpdateService.Java中的rebootAndUpdate函数中新建了一个线程，此此线程调用了**RecoverySystem.installPackage(mContext,mFile)**

按钮监听事件，调用mService.rebootAndUpdate（new  File(mFile)）。此处的file即为升级包

跟踪file来源，mService.getInstallFile() 获取到OTA在线下载的文件路径

注：从Bootloader开始如果没有组合键按下，就从MISC分区读取BCB块的command域，在主系统时已经将“boot-recovery”写入。然后进入recovery模式开始启动，与正常的启动系统类似，也有启动内核，然后启动文件系统，在进入文件系统后会执行/init，init的配置文件就是/init.rc。（bootable/recovery/etc/init.rc）

主要工作：

1. 设置环境变量
2. 建立etc连接
3. 新建目录，备用
4. 挂载/tmp为内存文件系统tmpfs
5. 启动recovery（/sbin/recovery）服务
6. 启动adbd服务（用于调试）

（\*其中最重要的就是启动recovery服务）

源码路径：bionic/libc/unistd/reboot.c

此函数将”recovery”传过去，之后将”boot-recovery”写入到BCB数据块的command域中，重启之后就知道进入recover

进入reboot：

调用pm.reboot(“recovery”)，跟踪发现其实是一个系统调用\_\_reboot(LINUX\_REBOOT\_MAGIC1,LINUX\_REBOOT\_MAGIC2,mode,NULL)，前两个参数代表组合键，mode代表传过来的”recovery”

**Recovery流程**

Recovery的内容有：恢复出厂设置、OTA升级、使能/关闭加密文件系统

（下图是recovery整个流程，很详细，来自http://blog.csdn.net/mu0206mu/article/details/7465439）

1、ui\_init():在recovery的过程中主要用于显示一个背景图片（正在安装或者安装失败）和一个进度条（用于显示进度）,另外还启动了两个线程，一个用于处理进度条的显示（progress\_thread），另一个用于响应用户的按键（input\_thread）。

2、 get\_arg()：这个函数主要做了上图中get\_arg()往右往下直到parse arg/v的工作。对照着流程看。

      ①get\_bootloader\_message()：主要工作是根据分区的文件格式类型（mtd或emmc）从MISC分区中读取BCB数据块到一个临时的变量中。

      ②然后开始判断Recovery服务是否有带命令行的参数（/sbin/recovery，根据现有的逻辑是没有的），若没有就从BCB中读取recovery域。如果读取失败则从/cache/recovery/command中读取然后。这样这个BCB的临时变量中的recovery域就被更新了。在将这个BCB的临时变量写回真实的BCB之前，又更新的这个BCB临时变量的command域为“boot-recovery”。这样做的目的是如果在升级失败（比如升级还未结束就断电了）时，系统在重启之后还会进入Recovery模式，直到升级完成。

      ③在这个BCB临时变量的各个域都更新完成后使用set\_bootloader\_message()写回到真正的BCB块中。

                这个过程可以用一个简单的图来概括，这样更清晰：



  3、parserargc/argv：解析我们获得参数。注册所解析的命令（register\_update\_command），在下面的操作中会根据这一步解析的值进行一步步的判断，然后进行相应的操作。

   4、 if(update\_package)：判断update\_package是否有值，若有就表示需要升级更新包，此时就会调用install\_package()（即图中红色的第二个阶段）。在这一步中将要完成安装实际的升级包。

     5、 if(wipe\_data/wipe\_cache)：这一步判断实际是两步，在源码中是先判断是否擦除data分区（用户数据部分）的，然后再判断是否擦除cache分区。值得注意的是在擦除data分区的时候必须连带擦除cache分区。在只擦除cache分区的情形下可以不擦除data分区。

     6、maybe\_install\_firmware\_update()：如果升级包中包含/radio/hboot firmware的更新，则会调用这个函数。查看源码发现，在注释中（OTA INSTALL）有这一个流程。但是main函数中并没有显示调用这个函数。目前尚未发现到底是在什么地方处理。但是其流程还是向上面的图示一样。即，① 先向BCB中写入“boot-recovery”和“—wipe\_cache”之后将cache分区格式化，然后将firmware image 写入原始的cache分区中。②将命令“update-radio/hboot”和“—wipe\_cache”写入BCB中，然后开始重新安装firmware并刷新firmware。③之后又会进入图示中的末尾，即finish\_recovery()。

     7、 prompt\_and\_wait()：这个函数是在一个判断中被调用的。其意义是如果安装失败（update.zip包错误或验证签名失败），则等待用户的输入处理（如通过组合键reboot等）。

     8、 finish\_recovery()：这是Recovery关闭并进入Main System的必经之路。其大体流程如下：



       ① 将intent（字符串）的内容作为参数传进finish\_recovery中。如果有intent需要告知Main System，则将其写入/cache/recovery/intent中。这个intent的作用尚不知有何用。

       ② 将内存文件系统中的Recovery服务的日志（/tmp/recovery.log）拷贝到cache（/cache/recovery/log）分区中，以便告知重启后的Main System发生过什么。

       ③ 擦除MISC分区中的BCB数据块的内容，以便系统重启后不在进入Recovery模式而是进入更新后的主系统。

        ④ 删除/cache/recovery/command文件。这一步也是很重要的，因为重启后Bootloader会自动检索这个文件，如果未删除的话又会进入Recovery模式。原理在上面已经讲的很清楚了。

     9、 reboot()：这是一个系统调用。在这一步Recovery完成其服务重启并进入Main System。这次重启和在主系统中重启进入Recovery模式调用的函数是一样的，但是其方向是不一样的。所以参数也就不一样。查看源码发现，其重启模式是RB\_AUTOBOOT。这是一个系统的宏。

**Install\_package详解**

源码路径：/bootable/recovery/install.c



①ensure\_path\_mount()：先判断所传的update.zip包路径所在的分区是否已经挂载。如果没有则先挂载。

②load\_keys()：加载公钥源文件，路径位于/res/keys。这个文件在Recovery镜像的根文件系统中。

 ③verify\_file()：对升级包update.zip包进行签名验证。

 ④mzOpenZipArchive()：打开升级包，并将相关的信息拷贝到一个临时的ZipArchinve变量中。这一步并未对我们的update.zip包解压。

  ⑤try\_update\_binary()：在这个函数中才是对我们的update.zip升级的地方。这个函数一开始先根据我们上一步获得的zip包信息，以及升级包的绝对路径将update\_binary文件拷贝到内存文件系统的/tmp/update\_binary中。以便后面使用。

   ⑥pipe()：创建管道，用于下面的子进程和父进程之间的通信。

   ⑦fork()：创建子进程。其中的子进程主要负责执行binary（execv(binary,args)，即执行我们的安装命令脚本），父进程负责接受子进程发送的命令去更新ui显示（显示当前的进度）。子父进程间通信依靠管道。

   ⑧其中，在创建子进程后，父进程有两个作用。一是通过管道接受子进程发送的命令来更新UI显示。二是等待子进程退出并返回INSTALL SUCCESS。其中子进程在解析执行安装脚本的同时所发送的命令有以下几种：

     progress  <frac> <secs>：根据第二个参数secs（秒）来设置进度条。

     set\_progress  <frac>：直接设置进度条，frac取值在0.0到0.1之间。

     firmware <”hboot”|”radio”><filename>：升级firmware时使用，在API  V3中不再使用。

      ui\_print <string>：在屏幕上显示字符串，即打印更新过程。

      execv(binary,args)的作用就是去执行binary程序，这个程序的实质就是去解析update.zip包中的updater-script脚本中的命令并执行。由此，Recovery服务就进入了实际安装update.zip包的过程。