Smali 怎么移植百度云 ROM 手册

文档修改记录		
修改时间	修改内容	Owner
2013/6/7	创建文件，初始化内容	唐柳湘
2013/6/13	增加如何 root，新建 smali 项目	张威平
2013/7/2	精简文档	唐柳湘

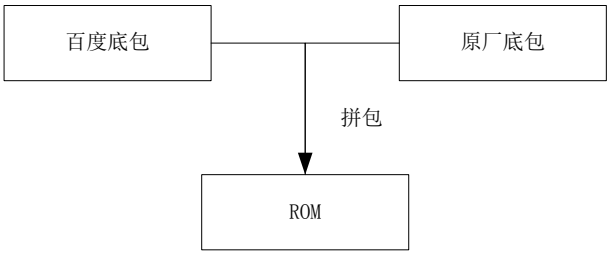
1. 百度云 ROM 制作原理

百度云 ROM 制作的原理很简单，可以用四个字来形容“拼包+插桩”。详情见后续章节。

1.1. 拼包

所谓的拼包就是将百度的底包和厂商的底包有选择性地拼在一起，制作一个新的 ROM。

在这里百度定义好了它修改或增加的文件（详情可见 build/configs/baidu_default.mk），然后用这些文件去覆盖厂商的文件，做成最后的 ROM。如下图所示：

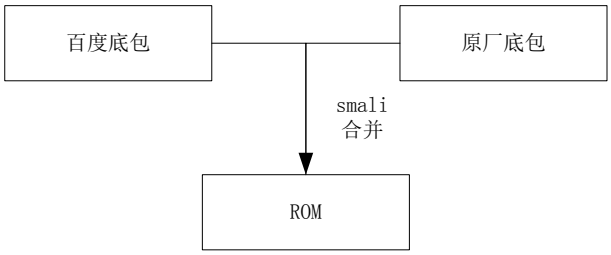


百度云 ROM 制作--拼包

但是如果完全按照拼包的方式制作百度云的 ROM，会导致系统的稳定性下降。因为 so 库等文件是无法修改的，一旦存在兼容性的问题，将无法解决。因而我们需要 smali 插桩的方式在 framework 层注入百度的功能，解决兼容性的问题。

1.2. 插桩

在 framework 层的 jar 包中，通过将百度自身的 feature 通过 smali 的方式插入原厂的 smali 代码中，从而让百度云 ROM 能够在手机上跑起来。



百度云 ROM 制作—smali 合并

详情参见机型文档及机型的提交记录。

1.3. 选择合适的百度底包

前面讲到了百度云 ROM 的制作原理，“拼包和插桩”，显然选择合适的百度和原厂的底

包是很重要的。如果不稳定，或差异太大，匹配起来的难度必然会很大，而且会很不稳定。

1.3.1. 百度云 ROM 机型参数

百度云 ROM 的机型的参数如下表所示，请尽量选择与自己手机相似的机型：

机型	Android 版本	网络类型	分辨率	网络模式	芯片厂商	CPU 型号	手机厂商
Nexus S	4.0	单卡双模	800x480(hdpi)	CDMA，GSM	三星	Samsung S5PC110	三星
I9300	4.0	单卡双模	1280x720(xhdpi)	CDMA，GSM	三星	三星 Exynos 4412	三星
A789	4.0	双卡双模	800x480(hdpi)	WCDMA，GSM	Mtk	MT6577	联想
A798T	4.0	单卡双模	854x480(hdpi)	TD-SCDMA，GSM	Mtk	MT6517	联想
C8813	4.1	单卡双模	854x480(hdpi)	CDMA，GSM	高通	MSM8625	华为

1.3.2. 选择百度 ROM 底包原则

选 base 机型原则如下，其中的比较都是跟原厂的底包进行的：

1、 Android 版本、网络类型必须一致，即单卡只能选单卡底包，双卡选双卡的

2、 分辨率：

a) 最好保持一致，如果不一致，请注意更新开关机动画

b) 实在不行，可以选同为 hdpi 或 xhdpi

3、 其他：

a) Mtk 上面网络模式没有影响，是全部都支持的，即在 mtk 上不需要考虑是移动、联通还是电信的 3G，只需要考虑单双卡即可。

b) 芯片厂商、CPU 型号、手机厂商：如果可以，尽量选择跟自己的一致的。

1.4. 选择合适的原厂底包

选择原厂底包的时候，android 的版本当然是很重要的，如原厂有 2.3 的版本、4.0 的版本，4.1 的版本，而能找到的适合移植的的百度底包只有 4.0 的，那我们也只能选择原厂 4.0 的底包。

一个稳定、bug 少的原厂底包，将会减少你的 ROM 的底层 bug。因为百度 ROM 的开发方式，决定了 framework 层以下的问题解决起来会比较麻烦。

故在选择原厂底包前，请在网上进行充分调研看这个版本的稳定性以及 bug 情况。

2. 准备

2.1. OS

装有 ubuntu 的 PC，或在 windows 上通过 vmware 安装 ubuntu
版本：ubuntu 10.04 以上

2.2. 需要的软件

需要安装 jdk、git、curl、repo 等软件，安装方法如下：

```
$ sudo apt-get install sun-java6-jdk
$ sudo apt-get install git-core curl
$ mkdir ~/bin
$ curl http://android.git.kernel.org/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

3. 构建百度云 ROM

3.1. 新建 smali 工程

新建 smali 工程的准备工作如下：

1、 root 手机，并连接上电脑
2、 在 devices 目录下新建一个目录，目录名为项目名称
3、 在该目录下准备好厂商的 boot.img，recovery.img（没有 recovery.img，可用 recovery.fstab 代替）
4、 参照附录中的 Makefile 详解配置好 Makefile
5、 然后执行 make newproject

编译成功后，目录结构如下：

```
├──── framework-res
├──── framework.jar.out
├──── secondary_framework.jar.out
├──── services.jar.out
├──── out
│      ├──── obj
│      ├──── oem_target_files.zip
│      └──── oem_target_files.zip.deodex.zip
└──── vendor
      ├──── BOOT
      ├──── BOOTABLE_IMAGES
      ├──── META
      ├──── METAINF
      ├──── OTA
      ├──── RADIO
      ├──── RECOVERY
      ├──── system
      └──── vendor.zip
```

3.2. 下载百度底包

在开始编译项目之前，需要从百度云 ROM 论坛下载最新的百度云 ROM 底包，具体的底包的选择请参考 2.4 章节所述。

将下载的底包放到工程目录 baidu 目录下(devices/xxx/baidu，需要自己新建)，并重命名为 baidu.zip，即 baidu/baidu.zip

论坛地址：

http://bbs.rom.baidu.com/download.php

3.3. 编译整个项目

编译整个项目的方法如下表所示：。

准备	命令	备注
下载百度 base 包 baidu.zip	make ota/otapackage/fullota 或 make	a) ota、otapackage、fullota 和 make 不带参数四种方法等价 b) 输出：out/ota_$PRJ_NAME.zip 比如 a789 编译完之后，输出为：out/ota_a789.zip

3.4. 编译单个目标

单个目标的编译，按照目标的类型分为以下几种。如果要编译完成后，自动 push 到手机上，请 make xxx.phone 即可（只支持 apk 和 jar 在编译完成后，自动 push 到手机上）。具体如下：

目标	命令	输出	备注
xxx.apk	make xxx	out/merged_target_files/SYSTEM/[app/framework]/xxx.apk	xxx 为 apk 的前缀，比如 Phone, BaiduCamera 等
xxx.jar	make xxx	out/merged_target_files/SYSTEM/framework/xxx.jar	xxx 为 jar 的前缀，比如 framework, android.policy 等
push 到手机	make xxx.phone	会先编译单个 apk 或 jar，在进行签名后，push 到手机上	只支持单个 apk 或 jar 包编译完成后 push 到手机上 例如： make Phone.phone，则会在编译完 Phone 之后，将 Phone.apk push 到手机上
boot.img	make bootimage	out/boot.img	会编译出最新的 boot.img
recovery.img	make recoveryimage	out/recovery.img	会编译出最新的 recovery.img

3.5. clean

可以使用的 clean target 有两种，clean 和 clean-all

目标	命令	备注
clean 单个工程	make clean	会删除 out、source 目录
clean 单个工程及下载的 base 包	make clean-all	会删除 out、source、以及下载的 base 包 这样子下次编译的时候，必须要去服务器重新下载 base 包
clean 单个目标	make	只支持单个 apk 或 jar 包的 clean，比如想要 clean

	clean-xxx	Phone 的编译，则 make clean-Phone 即可

4. Bring up（怎么让系统起机）

所谓的 Bring up 就是让百度云 ROM 能够在进到桌面，这个时候需要 merge 的基本的百度 feature 可参考机型文档。

在 bring up 之前，建议 policy 用原厂的，即在将原厂的 android.policy.jar 反编译放到工程目录下，并在 Makefile 里的 vendor_modify_jars 加上 android.policy。在 bring up 之后，换回用百度的（在 vendor_modify_jars 中去掉 android.policy），这个时候可能会少一些接口或函数。

附录

附录一：Makefile 配置详解

在新建项目的时候就需要配置好项目目录下的 Makefile，在配置时，可以参考 build/readme.txt 进行配置，或者基于一个相近的项目的 Makefile 进行修改，具体的可配置项如下：

编译选项配置

配置项	Values	作用
DALVIK_VM_BUILD	27/28/xx	目前只支持 27 和 28 两种 用于区分 libdvm.so 的版本，在做 odex 的时候会用到。
PLATFORM	qualcomm/mtk/sony	厂商的 boot.img、recovery.img 的解包和打包的方法，以及卡刷包里的 updater-script 书写形式区分
BASE_PLATFORM	qualcomm/mtk/sony	百度的 boot.img、recovery.img 的解包和打包的方法
PREBUILT_BOOT_IMG	true/false/空	true：不编译 boot.img，目标 boot.img 项目目录下的 boot.img false：编译 boot.img，会依赖于 vendor/BOOT/以及 source/BOOT 目录 空：目标里不包含 boot.img（慎用）
PREBUILT_RECOVERY_IMG	true/false/空	true：不编译 recovery.img，目标 recovery.img 项目目录下的 recovery.img false：编译 recovery.img，会依赖于 vendor/BOOT/以及 source/RECOVERY 以及 vendor/RECOVERY 目录 空：目标里不包含 recovery.img
BAIDU_FRAMEWORK_OVERLAY_TYPE	baidu-4.0/rom-mtk/rom-qc-4.1	用于选择当前 project 需要使用的 overlay 所属的分支 这个值的配置跟你选的百度底包有很大的关系，需要根据 android 的版本以及是否为 mtk 芯片作为参考，具体如下：

		Android 4.0 Mtk -> rom-mtk Android 4.0 非 mtk -> baidu-4.0 Android 4.1 -> rom-qc-4.1

各个 module 的配置

编译各个模块时的配置项如下,其中定制脚本指的是这个模块在编译的时候会调用的脚本，可以在这个脚本里进行一些特殊的定制。各个定制脚本的调用和参数的传递请参看“定制脚本”章节。

配置项	Values	作用	定制脚本
vendor_remove_dirs	相对于 system 的目录，如 app appbackup 等等	用于配置原厂的哪些目录结构需要去除，默认的配置在 vendor_default.mk 里配置	无
vendor_remove_files	相对于 system 的文件，如 t328t 里的 bin/zchgd 等等	用于配置原厂的哪些目录结构需要去除，默认的配置在 vendor_default.mk 里配置	无
vendor_saved_apps	原厂需要保留的 app，类似我们以前的 savedapp.list，但是需要去除 framework 里的 apk	原厂 app 目录下需要保留的 app，将采用原厂的签名类型进行签名	无
vendor_modify_apps	FMRadio	配置需要编译的 app eg: 比如需要替换原厂的 FMRadio 的 icon，则只需要将原厂的 FMRadio 反编译放到工程目录下，然后加上这个配置即可。	custom_app.sh
vendor_modify_jars	framework/secondary_framework/services/HTCExtension/xxx	用于编译 jar 包,base 是原厂的 jar 包，然后将我们的 feature 往上面进行 merge eg: 比如要将百度的 feature merge 到 framework.jar 中去，则需要将原厂的 framework.jar 反编译放到工程目录下，然后加上这个配置	custom_framework.sh
baidu_modify_apps	Phone/SystemUI/xxx	作用：base 的百度的 apk，然后往我们的 apk 注入资源，追加.smali.part	custom_app.sh

		eg1： 比如 t328t 需要往百度的 Phone 里面的 PhonInterfaceManager.smali 增加一个接口，则可以在 Phone/smali/com/android/phone/PhoneInterfaceManager.smali.part 文件里增加这个接口 eg2: 比如 t328t 需要覆盖 BaiduCamera 里面的 res/values/arrays.xml 里的某一个资源，则只需要在 BaiduCamera/res/values/arrays.xml 里配置即可	
baidu_modify_jars	android.policy/xxx	base 的百度的 jar 包，然后往百度的 jar 包里面追加.smali.part 文件 eg1: 比如 t328t，我们的 policy 少了几个接口，则可以在工程目录下建一个 android.policy.jar.out 目录，然后在对应的 package 下面，建立对应的.smali.part 文件	custom_framework.sh
override_property	xxx=xxx	会将 override_property 设置的属性写入 build.prop 中 Eg: override_property+= ro.baidu.default_write.settable=true 则会将 ro.baidu.default_write.settable=true 写入 build.prop 中去	custom_buildprop.sh
remove_property	xxx	会从 build.prop 中删除 remove_property 里配置的属性 eg1: remove_property += \ dev.defaultwallpaper 则会从 build.prop 中去除 dev.defaultwallpaper 属性	custom_buildprop.sh

附录二：常用工具使用

安装 framework 资源

注：安装 framework 资源是反编译 apk 前必须要做的工作！不然拿到的 res 会有问题

apktool: 安装单个 framework 资源

命令：apktool

用法：
apktool if framework-res.apk

作用：安装 framework-res.apk

ifdir:安装 framework 目录下所有资源

用法：ifdir FRAMEWORK_DIR
1、 FRAMEWORK_DIR：资源 apk 所在的目录

作用：安装 FRAMEWORK_DIR 目录下所有的资源 apk

Eg:
ifdir source/system/framework
会安装 source/system/framework 下所有的资源 apk

decode_all.sh:反编译目录下所有的 apk 或 jar

用法：decode_all.sh IN_DIR [OUT_DIR]
1、 IN_DIR：apk 或 jar 所在的目录
2、 OUT_DIR：反编译得到的 smali 的目标目录，如果为空，则为当前目录

作用：会找出 IN_DIR 目录下的所有的 apk 和 jar，然后调用 apktool 反编译到 OUT_DIR 目录

Eg：
decode_all.sh source/system/framework/ baidu_framework
则会将 source/system/framework/所有的 apk 和 jar 包，通过 apktool 反编译到

baidu_framework 目录

deodex.sh: 去除 odex

用法：deodex.sh ODEX.zip

1、 ODEX.zip：需要 odex 的 zip 包

作用：去除 zip 包的 odex

Eg:

deodex.sh ota-full_ns_201306180751.zip

会将 ota-full_ns_201306180751.zip 去除 odex

输出：ota-full_ns_201306180751.zip.deodex.zip

idtoname.py:资源 id 转换为#type@name#[ta]

用法：idtoname.py PUBLIC_XML SMALI_DIR

1、 PUBLIC_XML：framework-res 里的 public.xml

2、 SMALI_DIR：smali 文件所在目录

作用：将 smali 代码里的资源 id 转换为#type@name#[ta]的方式

Eg:

idtoname.py framework-res/res/values/public.xml framework.jar.out

会将 framework.jar.out 目录下所有的 smali 文件里的资源 id 转换为 #type@name#[ta]的方式。

注：这里的 framework-res 必须与 framework.jar.out 对应，不然会出现不一致的情况。

nametoid.py:将 #type@name#[ta]转换为资源 id

用法：nametoid.py PUBLIC_XML SMALI_DIR

1、 PUBLIC_XML：framework-res 里的 public.xml

2、 SMALI_DIR：smali 文件所在目录

作用：将 smali 代码里的资源#type@name#[ta]转换为 id 的方式，与 idtoname.py 相反

Eg:

nametoid.py framework-res/res/values/public.xml framework.jar.out

会将 framework.jar.out 目录下所有的 smali 文件里的#type@name#[ta]转换为资源 id。

解包 boot.img、recovery.img

目前支持的解包 mtk、sony 以及通用的 boot.img 和 recovery.img。具体如下：

命令	对应的平台	作用
unpack_boot_mtk	mtk	解包 mtk 平台的 boot.img，如果你的手机的芯片是 mtk 的，则用这个命令解包 boot.img
unpack_boot_sony	Sony 的 Xperia	解包 sony 的 Xperia 系列的 boot.img
unpack_boot	通用	Android 通用的解包 boot.img 方法（mtk 和 sony 的 Xperia 比较特殊）
unpack_rec_mtk	mtk	解包 mtk 平台的 recovery.img
unpack_rec	通用	Android 通用的解包 recovery.img 方法（mtk 和 sony 的 Xperia 比较特殊）

打包 boot.img、recovery.img

目前支持的打包 mtk、sony 以及通用的 boot.img 和 recovery.img。具体如下：

命令	对应的平台	作用
pack_boot_mtk	mtk	打包 mtk 平台的 boot.img，如果你的手机的芯片是 mtk 的，则用这个命令解包 boot.img
pack_boot_sony	Sony 的 Xperia	打包 sony 的 Xperia 系列的 boot.img
pack_boot	通用	Android 通用的打包 boot.img 方法（mtk 和 sony 的 Xperia 比较特殊）
pack_rec_mtk	mtk	打包 mtk 平台的 recovery.img
pack_rec	通用	Android 通用的打包 recovery.img 方法（mtk 和 sony 的 Xperia 比较特殊）

快速 cd 命令

在执行完 source ./build/envsetup.sh 之后，会生成以下命令：

croot：会 cd 到 smali 的根目录

假如有项目 onex，则 conex 会 cd 到 onex 的项目目录。

定制脚本的作用

custom_app.sh

作用

在编译 app、framework 里的 apk 的时候会调用，主要是为了方便对单个 apk 的编译进行定制。

参数

1、 apk 的 basename：比如 Phone.apk 对应的则为 Phone

2、 smali 代码的路径：用 apktool 反编译得到的 smali 代码的路径

cusotm_jar.sh

作用

在编译 framework 里的 jar 的时候会调用，主要是为了方便对单个 jar 的编译进行定制。

参数

1、 jar 的 basename：比如 framework.jar 对应的则为 framework

2、 smali 代码的路径：用 apktool 反编译得到的 smali 代码的路径

custom_targetfiles.sh

作用

在打包成 target-files.zip 前会被调用，可以在 custom_targetfiles.sh 里面去除或增加一些文件等等。

参数

无

附录三：常见问题

mtk 平台怎么从手机上获取 boot.img

第一步、将 su 放入 system 分区

1）刷入第三方 recovery；

2）如果有第三方“root 破解 ota 包”，可以刷入“root 破解 ota 包”；

一般第三方 root 破解包都是通过在 system 分区放入 su 和 super.apk 来实现 root；

3）如果没有第三方“root 破解 ota 包”，那么就需要手动将 su 放进 system 分区；

解压 recovery.img，在 RAMDISK/etc/recovery.fstab 可以找到 system 分区挂载点；

```
# mount point     fstype              device                          [device2]
/system             ext4            /dev/block/mmcblk0p5
```

手机在 recovery 状态下，将手机与 PC 连接，adb shell 进入手机，挂载 system 分区，

```
ubuntu:/$ adb shell
android:/# mount /dev/block/mmcblk0p5 /system
```

将准备好的 su push 到手机上

```
ubuntu:/$ adb push su /system/bin/
ubuntu:/$ adb shell chmod 4755 /system/bin/su
```

4）重启手机，进入系统。

第二步、通过 su 得到 boot 分区

1）adb shell 进入手机

2）执行 su 命令，获取超级用户权限

```
ubuntu:/$ adb shell
android:/$ su
android:/#              [如果此时命令提示符变成“#”，则表明 su 成功]
```

3）找到 boot.img 分区

cat proc/dumchar_info 查看手机分区表

```
android:/# cat proc/dumchar_info
Part_Name              Size                StartAddr         Type      MapTo
bootimg    0x0000000000600000  0x0000000000988000   2    /dev/block/mmcblk0
recovery   0x0000000000600000  0x0000000000f88000   2    /dev/block/mmcblk0
```

可以看到 boot.img 是从 0x988000 地址开始到 0xf88000 结束，大小为 0x0600000

4）将 boot.img 分区 cat 到 sdcard 上

cat 整个分区时间会很长，执行了 5 秒后"Ctrl + C"切断即可

```
android:/# cat /dev/block/mmcblk0 > /mnt/sdcard2/block_file
```

5）将取得的 boot.img 分区，按分区地址裁剪

```
ubuntu:/$ adb pull /mnt/sdcard2/block_file .
ubuntu:/$ head -c 16285696    block_file > head_file
      [16285696    <- 0xf88000  不支持十六进制，转成十进制]
ubuntu:/$ tail -c 6291456 head_file > boot.img
      [6291456    <- 600000  不支持十六进制，转成十进制]
```

6）解压 boot.img

如果能成功解压 boot.img 得到 kernel 和 RAMDISK，则表明提取 boot.img 成功

no rule to make target xxx/baidu.zip

错误日志：

make: *** No rule to make target `/xxx/baidu.zip', needed by `/xxx/baidu.deodex.zip'. Stop.

make: *** Waiting for unfinished jobs....

解决办法：

从百度云 ROM 论坛下载百度的 ota 包放在工程目录下的 baidu 目录，并命名为 baidu/baidu.zip

怎样修改百度 apk 的资源

例如我要修改 BaiduCamera.apk 的资源，则需要按以下步骤操作即可：

1、 在 Makefile 中增加以下配置：

baidu_modify_apps += BaiduCamera

2、 然后在项目目录下创建目录 BaiduCamera/res

mkdir –p BaiduCamera/res

3、 然后 BaiduCamera/res 增加对应需要替换 BaiduCamera.apk 的资源

怎样修改厂商 apk 的资源

例如我要修改厂商的 FMRadio.apk 的图标，则需要按以下步骤操作即可：

1、 在 Makefile 中增加以下配置：

vendor_modify_apps += FMRadio

2、 按以下操作，反编译 FMRadio.apk：

```
在项目的根目录下执行：
$ ifdir vendor/system/framework/
$ apktool d vendor/system/app/FMRadio.apk
```

3、 根据 android:icon 的值，找到对应的 icon 图片，然后替换即可

厂商的 apk 或 jar 缺少函数

解决办法：

直接在反编译得到的 smali 代码里添加即可。

百度的 apk 或 jar 缺少函数

比如 Phone

解决办法：

/main.mk: No such file or directory

请去 smali 根目录执行 source ./build/envsetup.sh

怎样往 system 增加文件

可以将文件放在项目目录下的 overlay/system/，则会在编译完成后拷贝到 out 里面去

怎样修改 init.rc

在项目目录下的 vendor/BOOT/RAMDISK/init.rc 中修改

拆分 jar 包

什么时候需要拆分 jar 包？

当编译的时候出现“method index is too large”，或运行时出现“LinearAlloc exceeded capacity”则需要拆分对应的 jar 包。

编译时的错误：

```
Exception in thread "main" org.jf.dexlib.Util.ExceptionWithContext: method index is too large.
        at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
        at org.jf.dexlib.Item.addExceptionContext(Item.java:177)
        at org.jf.dexlib.Item.writeTo(Item.java:120)
        at org.jf.dexlib.Section.writeTo(Section.java:119)
        at org.jf.dexlib.DexFile.writeTo(DexFile.java:716)
```

```
        at brut.androlib.src.DexFileBuilder.getAsByteArray(DexFileBuilder.java:75)
        at brut.androlib.src.DexFileBuilder.writeTo(DexFileBuilder.java:58)
        at brut.androlib.src.SmaliBuilder.build(SmaliBuilder.java:50)
        at brut.androlib.src.SmaliBuilder.build(SmaliBuilder.java:35)
        at brut.androlib.Androlib.buildSourcesSmali(Androlib.java:243)
        at brut.androlib.Androlib.buildSources(Androlib.java:200)
        at brut.androlib.Androlib.build(Androlib.java:191)
        at brut.androlib.Androlib.build(Androlib.java:174)
        at brut.apktool.Main.cmdBuild(Main.java:185)
        at brut.apktool.Main.main(Main.java:70)
Caused by: java.lang.RuntimeException: method index is too large.
        at org.jf.dexlib.Code.Format.Instruction35c.writeInstruction(Instruction35c.java:102)
        at org.jf.dexlib.Code.Instruction.write(Instruction.java:57)
        at org.jf.dexlib.CodeItem.writeItem(CodeItem.java:258)
        at org.jf.dexlib.Item.writeTo(Item.java:117)
        ... 12 more
code_item @0x1d2d18 (Lorg/codeaurora/Performance;->cpuBoost(I)V)
```

运行时的错误：

```
E/dalvikvm( 7815): LinearAlloc exceeded capacity (5242880), last=3906240
```

怎样拆分 jar 包？

拆分 jar 包的方法比较简单，可以参考已有机型的拆分方法。

比如需要拆分 framework.jar 成 framework.jar 和 secondary_framework.jar（名字可以随便取，但是需要跟 init.rc 的 BOOTCLASSPATH 对应）

1、 在 Makefile 中的 local_build_jars 加上 secondary_framework

2、 在工程目录创建 secondary_framework.jar.out，将 framework.jar.out 里的 apktool.yml 拷贝到 secondary_framework.jar.out。将 apktool.yml 里的 apkFileName 改为 secondary_framework.jar

3、 将 framework.jar.out 里面的 package，比如 smali/android/app，smali/android/view 等等（根据自己的需要定，没有特殊的要求），将这些 package 从 framework.jar.out 挪到 secondary_framework.jar.out，当然这些 package 还得保留在 framework.jar.out 的目录结构。

4、 修改 init.rc 的 BOOTCLASSPATH，在你被你拆分的 jar 包后加上新的 jar 包。比如这里则需要在 framework.jar 后加上 secondary_framework.jar