ColorOS Smali 移植指南 v0.1

更改记录			
版本	更改日期	更改内容	修订人
V0.1	2013-08-20	创建	Smali 适配团队

目　录

1 移植前的准备

1.1 选择合适的目标 ROM

目前提供的 ColorOS 是基于 4.2 google 源码开发的，第三方机型 4.2（除了 MTK 平台）的原厂 ROM 版本都是合适的。

1.2 刷机

1.2.1 第一次刷机

以三星 S4 为例，将制作好的 zip 升级包 push 到 S4 的 SD 卡，通过 adb shell 命令到 sd 卡中查看 update.zip 包的大小是否一致保证 zip 包完整。如果确认无误后关机，进入下一步操作。

确认关机后，同时按住电源键、HOME 健和音量（-）键进入 fastboot，插上 USB 进行烧录 boot 和 recovery。

fastboot.exe 是 android sdk 自带的烧录工具

1、 烧录针对 S4 修改的 boot.img

命令：fastboot flash boot fastboot flash boot out\boot.img（boot.img 文件）

2、 烧录针对 S4 修改的 recovery.img

命令：fastboot flash recovery　out\recovery.img（recovery.img 文件）

3、 进入 recovery

命令：fastboot boot out\recovery.img　（recovery.img 文件）

4、 选择 zip 包进行升级。

进入 recovery 后选，先选择语言(简体中文)->（清除数据和缓存）->(清除所有数据)->确定->(清除数据和缓存）->(从 SD 卡安装)。 选择 push 到 SDK 的 zip 升级包进行升级。

备注：第一次刷机一定要先烧录 boot，然后烧录 recovery，否则升级成功后 recovery 会被还原为三星的 recovery。这样每次升级都需要烧录 recovery.img

1.2.2 如何进入 recovery

除了 1.2.1 中第 3 条通过命令可以进入 recovery 外还可以通过电源键 + HOME 健 + 音量上键进入 recovery。

关机后，同时按住电源键、HOME 键和音量（+）键，待出现三星 LOGO 时放开电源键保持 HOME 键不放开，便可进入 recovery。

1.3 解包与打包

1.3.1 解包

将编译好的 jar 包或 odex 文件转换为 smali 文件是通过开源工具 baksmali.jar 进行反编译。

baksmali.jar

为了让反编译生成的 smali 文件格式保持一致，在进行反编译时需加上”-l”参数。

例如：

```
java -jar tools/baksmali.jar -a 16 -l -x jar/framework_ext.odex -o out/framework_ext -d jar
```

具体的参数说明可以运行 java -jar tools/baksmali.jar –help 查看。

1、 反编译 odex 文件。

反编译 odex 文件需要加上-x 参数，同时要包含于该文件相关联的 odex 文件才能进行。例如：

java -jar tools/baksmali.jar -a 16 -l -x jar/services.odex -o out/services -d jar

在 jar 目录下需要有 android.policy.odex、bouncycastle.odex、core.odex、core-junit.odex、ext.odex、framework.odex 和 secondary-framework.odex。

2、 反编译非 odex 文件。

非 odex 文件直接反编译即可，例如：

java -jar tools/baksmali.jar -a 16 -l smali/oppo-framework.jar -o out/oppo-framework

1.3.2 打包

将修改好的 smali 文件重新打包为 jar 是通过 smali.jar 来完成。


smali.jar

命令如下：

java -jar tools/smali.jar -a out/framework -o out/classes.dex

然后切换到 out 目录，用 jar 将 classes.dex 打包为 framework.jar

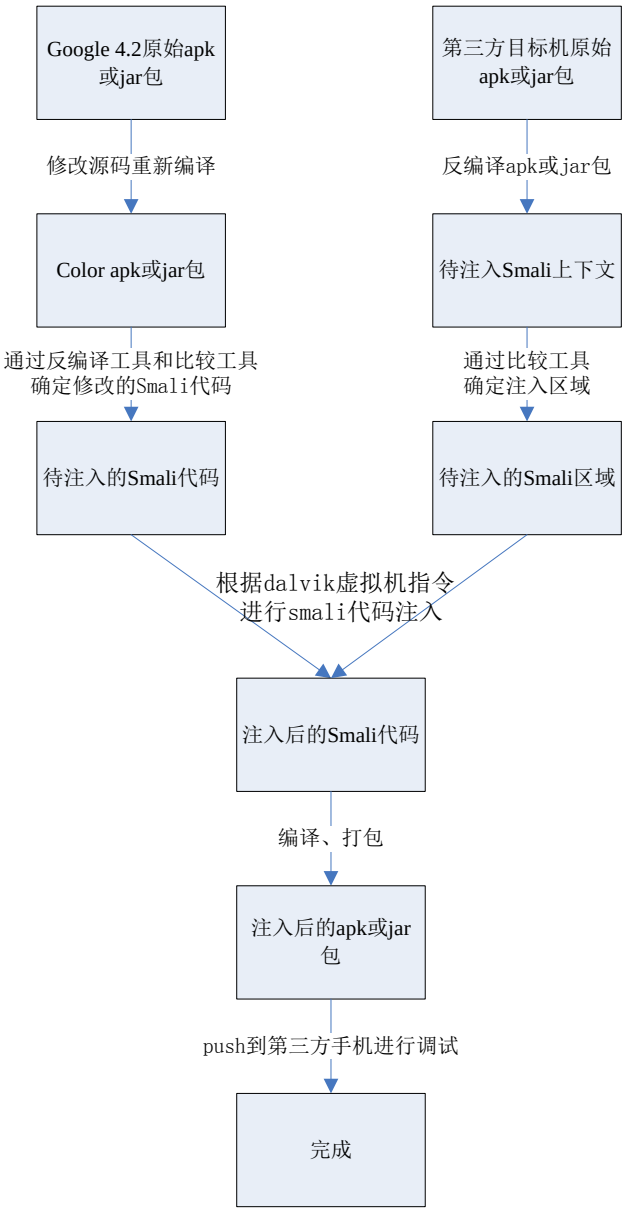
jar -cvf framework.jar classes.dex

将生成的 framework.jar push 到手机重启手机即可验证 smali 移植。

2 移植流程

移植 Color os 过程大致分为五步——确定需要注入的 Samli 代码，确定注入位置，注入 Smali 代码，编译 Smali 代码，调试 Smali 代码。

总体流程如下图：



1、确定需要注入的 Smali 代码

首先，确定基线文件——Google 源码的 apk 包或 jar 包，使用 apktool 反汇编，生成原始的 Smali 文件。

其次，修改对应 Color 的 apk 包或 jar 包的 java 源代码，使用编译系统重新生成新的 apk 包或 jar 包，并用 apktool 反汇编，生成包含修改后的 Smali 文件。

最后，使用比较工具（例如 BeyondCompare）比较两次 Smali 文件，即可提取出需要注入的 Smali 代码。

2、确定注入位置

首先，使用 apktool 反汇编第三方目标机的 apk 包或 jar 包，生成 Smali 文件。

其次，使用比较工具（例如 BeyondCompare）与基线文件进行对比，比较两次

Smali 文件，确定需要注入 Smali 位置。

3、 注入代码

首先，将 Color 的 Smali 代码注入对应的第三方目标机代码区域。

其次，对注入的 Smali 代码进行本地修改——修改变量、跳转标号、逻辑判断标号等，使之符合当前的 Smali 代码实现，完成“嫁接”工作。当然，如果情况很复杂，需要重写对应的 Smali 代码或者重构 java 源代码，来完成最终的代码注入。

4、 编译和调试

1）使用组合命令即可将 Smali 代码编译成 jar 包。

java -jar tools/smali.jar -a out/framework -o out/classes.dex

java -Xmx512M -jar smali.jar framework -o classes.dex

2）调试方法

命令:

adblogcat | grep dalvikvm

Dalvikvm: 给出调用栈，上下文执行过程。

adblogcat | grep VFY

VFY:会给出 Smali 代码出错的文件、函数以及错误原因。

3）常见运行错误及分析思路

➢函数变量列表与声明不同

A 调用时传入类型与声明不一致

B 调用时参数个数与声明不一致

➢函数调用方式不正确

A public 和 package 级别的方法用:invoke-virtual

B private 级别的方法用: invoke-director

C interface 级别的用: invoke-interface

➢类接口没有实现

添加父类接口空实现即可。

➢签名不正确

adb logcat | grep mismatch 查看前面错误的 package。

➢资源找不到

A 系统资源 apk 包签名异常，导致找不到系统资源。

B Smali 代码中资源 ID 错误，导致无法在系统资源找到。

C 资源相关类移植异常，导致资源加载异常。

3 工具和资源

3.1 工具

apktool	反编译 framework-res.apk
aapt	重新打包 framework-res.apk
baksmali.jar	将 odex 转换为 smali
dex2jar.sh	将 jar 包转换为 java 字节码
smali.jar	将 smali 转换为 dex 文件
signapk	签名工具
打包 bootimage 工具	minigzip
	mkbootfs
	mkbootimg
split_bootimg.pl	boot 解包工具
patch_color_framework.sh	将 color 与原版的差异 patch 到目标机型的 framework

使用方法：patch_color_framework.sh google_framework color_framework target_framework

3.2 资源

1、 apktool 进行反编译 framework-res.apk。

2、 将 oppo 针对原生资源的修改移植到 framewor-res 中

3、 用 aapt 命令进行重新打包 framework-res.apk

./tools/aapt p -f -m -x -z -J gen -S framework-res/res -M

framework-res/AndroidManifest.xml -F framework-res.apk -P gen/public_resources.xml -A framework-res/assets/

在没有合入 oppo 修改资源前最好重新打包 framework-res.apk，并备份 gen 目录下生成的内容，用于对比合入资源修改的生成文件 public_resources.xml，这样可以清晰的看到新增资源分配的 ID。

将目标机型的 framework.jar（如果是 odex，需要用 baksmali.jar 转换为 smali，然后通过 smali.jar 转换为 dex 文件），用 dex2jar 工具转换为 java 字节码，用 jd_gui 打开转换后的文件。

用 jd_gui 文件打开后，将 android.R.、android. Manifest 和 com.android.internal.R 拷贝替换 framework/base/res/java 目录下对应的 Java 文件内容。这些 Java 代码将编译到用于 smali 移植的 framework.jar 中。

4、 基于重新生成的 framework-res.apk 编译 oppo-framework-res.apk

./tools/aapt p -f -m -x -z -J gen-oppo -S oppo-res/res -M oppo-res/AndroidManifest.xml -F oppo-framework-res.apk -P gen-oppo/public_resources.xml -A oppo-res/assets/ --oppo-package 12 --oppo-public-id 1024 -I framework-res.apk

5、 对新生成的两个资源包进行签名

java -jar tools/signapk.jar tools/oppo-security/platform.x509.pem tools/oppo-security/platform.pk8 framework-res.apk ../out/framework-res.apk

java -jar tools/signapk.jar tools/oppo-security/platform.x509.pem tools/oppo-security/platform.pk8 oppo-framework-res.apk ../out/oppo-framework-res.apk

6、 执行 rom_build 目录下的./build-jar.sh 进行编译 java 代码，编译成功后结果会输出到 out 目录。

7、 解压 boot.img 修改 init.rc 框架

4 移植 ColorOS Framework

4.1 ColorOS

Color 下有两个子目录 src 和 system。src 存放的是 Color 修改过的代码。其中，值得注意的是在 4.2 版本上 Telephony 部分代码从 src\frameworks\base 路径转移到

src\frameworks\opt 路径下。在编译时新增了一个 telephony-common.jar 包。

src/frameworks/color 目录下放的是 Color 对原生资源的修改，有直接替换的图片，还有一些 xml 文件的改动。

system 分为四个子目录，目录结构映射手机中的 system 目录，也就是说这些目录中的文件最终会放到手机相应的目录中：

- system/framework ：存放由Color源码编译而成的android.policy.jar，framework.jar，secondary-framework.jar，services.jar，pm.jar，telephony-common.jar以及Color专属包oppo-framework.jar。
- system/app：存放Color核心应用
- system/media/theme/default：存放桌面默认主题，一共有四个文件（allApps.xml，allAppsBackup.xml，com.oppo.launcher 和 icons）
- system/xbin：busybox 程序和 Color 专属的 invoke-as 程序。

在移植的时候不要修改 Color 目录。

4.2 三星 S4

S4 是我们会移植三星 S4 机型新建的一个目录，每个机型都需要新建一个目录。该目录的组织规范为：

1）放置一个原厂 ROM 刷机包，比如 I9500-update.zip。

2）基于原厂 ROM 中的文件反编译修改的，需要存放反编译后整个目录的内容。比如 android.policy，framework，secondary-framework，services，pm，telephony-common。这些目录相应的是从刷机包中 system/framework/android.policy.jar ，system/framework/framework.jar，system/framework/services.jar ， system/framework/pm ， system/framework/telephony-common.jar 这些文件反编译而来的。

3. 移植资源

移植资源就是修改 framework-res.apk，先反编译原厂 ROM 中的 framework-res.apk，然后根据 src/frameworks/Color 目录下的文件，修改反编译 framework-res 目录中的相应文件。

4. 修改 Smali

修改 Smali 主要依据 5 大方法，即比较差异、直接替换、线性代码、条件判断、逻辑推理。

5 移植 ColorOS APP

所有需要移植的 ColorOS app 放在/system/app 目录下。移植 app 很简单，直接把对应的 apk 放到 system 分区 app 目录下即可。

ColorOS APP 总表

APK	说明
Phone.apk	电话
Browser.apk	浏览器
Contacts.apk	电话本
ContactsProvider.apk	电话本数据库
Email.apk	电子邮件
Mms.apk	短信
PhoneNOAreaInquireProvider.apk	电话归属地
TelephonyProvider.apk	短信相关数据库
Email.apk	电子邮件
OppoLauncher.apk	桌面
OppoLauncherSystem.apk	桌面辅助功能
SystemUI.apk	状态栏
OppoLockScreenManager.apk	锁屏管理
OppoLockScreenGlassBoard.apk	翻转解锁
OppoLockScreenTravel.apk	旅行解锁
OppoLockScreenWeather.apk	天气解锁
OppoLockScreenCard.apk	卡片解锁
OppoPasswordUnlock.apk	密码解锁
OppoPatternUnlock.apk	图案解锁
OppoSimUnlockScreen.apk	SIM 卡解锁
OppoDigitalClockWidget.apk	时钟 widget
OppoCalendarWidget.apk	日历 widget
OppoTimeWeatherWidget.apk	时钟天气 widget
OppoPrivateMoodAlbum.apk	心情相册专属页面
OppoPrivateMusicPage.apk	心情音乐专属页面
PackageInstaller.apk	程序安装
OppoLivepaper.apk	动态壁纸
OppoSimUnlockScreen.apk	SIM 卡解锁
OppoDigitalClockWidget.apk	时钟 widget

OppoCalendarWidget.apk	日历 widget
OppoTimeWeatherWidget.apk	时钟天气 widget
OppoPrivateMoodAlbum.apk	心情相册专属页面
OppoPrivateMusicPage.apk	心情音乐专属页面
PackageInstaller.apk	程序安装
OppoLivepaper.apk	动态壁纸
Calculator.apk	计算器
Calendar.apk	日历备忘录
CalendarProvider.apk	日历备忘录数据库
Clock.apk	时钟
FileManager.apk	文件管理器
OppoCompass.apk	指南针
OppoLockNow.apk	一键锁屏
OppoOTA.apk	OTA
OppoUsbSelection.apk	USB 选择界面
OppoWeather.apk	天气
OppoWeatherLocationService.apk	天气数据库
PowerManager.apk	省电管家
Settings.apk	设置
SettingsProvider.apk	设置数据库
MediaProvider.apk	多媒体数据库
NewSoundRecorder.apk	录音
OppoGallery2.apk	相册
OppoMusic.apk	音乐
VideoGallery.apk	视频

6 事项

1、在 smali 文件查找 OppoHook 即可找到 OPPO 修改代码的大致位置。

2、import 语句在 smali 文件是没有的，直接在实现中引用了头文件，因此移植时可以不管 import 的移植。

3、与 aidl 对应的 smali 文件一般有多个，移植时要注意对齐并确认是 ROM 加的接口。

4、新增文件的 smali 文件可以直接复制过去。

5、Phone.apk 具有平台区分性。高通平台、MTK 平台以及其它平台，Color 会提供三套不同的 Phone.apk，这三套 Phone.apk 是不能共用的，这点需特别注意。

6、先移植系统功能，最后再来用 APK 验证。最好能够按功能来做，移植好系统

功能，然后再放 APK 上去检查是否功能完善。

7、如果遇到以 dalvik 开头，后面跟着 VFT 这样的错误导致不能开机，一般都是寄存器使用出了问题，可以根据后面提示，到相应位置检查寄存器使用是否正确。

8、解锁移植的时候需要先将 OppoLockScreenManager 移植上去，其他解锁才可以正常运行。

9、Widget 的移植需要先将依赖的 APK 移植成功，Widget 的功能才能正常。

10、状态栏移植需要系统资源、通信服务和相关 StatusBar 服务移植完成，才能正常运行。

11、不要一次性移植完所有文件后再编译。如果插入代码有错误，会导致编译出错。而编译出错信息是天书，你无法知道是修改了哪个文件导致的错误。