ColorOS 适配教程

更改记录			
版本	更改日期	更改内容	修订人
V0.1	2014-03-13	创建	Circle Lee

目　录

1 PatchRom 使用介绍

1.1 选择合适的 ROM 包

市面上的手机都有着众多的 ROM，我们应该如何选择一个合适的 ROM 包作为基础包呢？

目前提供的两套 ColorOS 分别基于 4.2 Google 源码和 MTK6589 平台开发，第三方机型 4.2 的原厂 ROM 版本都是合适的。

原厂 ROM 版本虽然稳定性高，但是通常情况下原厂都会对自己的 ROM 进行大规模的修改。对于开发者而言，适配难度较大。因此开发者可以选择更接近 Google 源码的 CM 或者 AOKP 的包作为底包。当然，所选择的 ROM 包功能要尽量完善，稳定性尽量的高。

1.2 使用 PatchRom

ColorOS 基于不同平台发布两款不同的 PatchRom 脚本：color_2.0_patchrom_for_4.2_[版本号].zip 和 color_2.0_patchrom_for_mtk4.2_[版本号].zip。

color_2.0_patchrom_for_4.2_[版本号].zip 基于 AOSP 开发，适用于非 MTK 平台。color_2.0_patchrom_for_mtk4.2_[版本号].zip 基于 MTK 6589 平台开发，适用于 MTK 6589、6572、6582 平台。MTK 6592 平台代码差异较大，用该 PatchRom 适配难度大，目前暂不支持。

build: 该目录存放 ColorOS 特色 APK，资源文件，库文件，smali 代码以及使用到的工具。

tools: 该目录存放一些工具和脚本，在订制 ROM 和编译过程中需要使用这些工具和脚本。

device: 第三方 ROM 包放在该目录下。执行完脚本会打包生成一个第三方订制的 ColorOS ROM 包。

PatchRom 使用步骤：

1．将升级包命名为 update.zip 放在 device 目录下；

2．在当前目录执行". build/envsetup.sh"；

3. "cd device;make firstpatch"，根据 device 目录下的 temp/reject 文件，在 device/smali 路径下修改插桩失败文件；

4．对于非 MTK 平台机器，如果 smali/framework.jar.out 文件夹太大会导致打包失败，需要进行手动分包；

5．在 device 目录下执行"make fullota"，在 device 目录下生成 color-update.zip 就是生成的 color 升级包。

注意事项：

1．修改 boot.img，默认打开 USB 调试，加载 oppo-framework.jar。

2．对于非 MTK 平台机器，当第三方 ROM 包包含两个 framework.jar 时（例如 ROM 包包含 framework.jar 和 framework2.jar 情况），需要修改 device/makefile 里的两个变量：

ORGIN_SECOND_FRAMEWORK_NAME: 填入欲编译机器第二个 framework 文件名；

COLOR_FRAMEWORK_JARS: 填入欲编译机器的 framework 文件，主要看是否有第二个 framework。

3．对于非 MTK 平台机器，如果需要分包，请在 smali 文件夹下新建 XX.jar.out 格式的文件夹（例如：second-framweork.jar.out），并手动进行分包；

4．"device/custom-update":定制的文件夹，make fullota 时会直接覆盖过去，所以文件夹结构需要和升级包的保持一致，里面放一些原版不可删除的 system/app/下面的 apk 或者自己新增或修改的一些文件；

5. 如果获取到的升级包是 odex 的，需要先使用 deodex 工具合并，默认使用 apilevel15 合并命令如下：

deodex.sh update.zip

如果执行出错，请尝试指定 apilevel 值，如下：

deodex.sh -a 17 update.zip

1.3 修改 boot.img

内核 root 的关键是根文件系统中 default.prop 文件的两个属性 ro.secure 和 ro.debuggable 的值。根文件系统和内核一起放在 boot 分区中，如果我们能够修改 boot 分区中的这个文件，那么我们就可以自己 root 内核了。

一般来说某个机型的完整刷机包下有一个 boot.img 文件，该文件就是 boot 分区的镜像文件，安装刷机包时，会使用该文件刷写 boot 分区。Google 给 boot.img 文件定义了一个标准的格式，如果遵从这个标准格式，我们可以用下面的办法来修改它，但是如果不遵从，需要逛论坛详细的了解如何修改 boot 分区。

我们在patchrom 目录下，给定一个boot.img，输入命令解压boot.img：

tools/bootimgtools/split_bootimg.pl boot.img

解压后会看到一个 boot.img-ramdisk.gz 文件，该文件即是根文件系统的压缩包。还有一个 boot.img-kernel 文件，该文件即是 Linux 内核。

创建一个新的名叫 ramdisk 的目录，用于存放 ramdisk 盘中的文件。然后，提取出 ramdisk 文件。命令如下：

mkdir ramdisk

cd ramdisk

gzip -dc ../boot.img-ramdisk.gz | cpio –i

ramdisk 目录即为手机启动后的根文件系统目录，用任何编辑器修改 default.prop 文件和 init.rc 文件。

default.prop 修改 ro.secure 和 ro.debuggable 的值为：

ro.secure=0

ro.debuggable=1

init.rc 修改导入 oppo-framework.jar。如果对 framework.jar 进行了手动分包，此处也需要导入。

export BOOTCLASSPATH /system/framework/core.jar:/system/framework/core-junit.jar:/system/framework/bouncycastle.jar:/system/framework/ext.jar:/system/framework/framework.jar:/system/framework/oppo-framework.jar:/system/framework/telephony-common.jar:/system/framework/mms-common.jar:/system/framework/android.policy.jar:/system/framework/services.jar:/system/framework/apache-xml.jar

修改完成后开始打包

tools/bootimgtools/mkbootfs ramdisk | gzip > ramdisk-new.gz

tools/bootimgtools/mkbootimg --kernel boot.img-kernel --ramdisk ramdisk-new.gz -o oppo-boot.img

然后，复制到刷机目录中，替换原来的 boot.img。oppo-boot.img 需要重命名为 boot.img。

1.4 修改 build.prop

必须在 build.prop 文件中按照要求进行相应的修改。修改版本号，添加作者名和渠道。为了 OPPO 能够进行统计，以更好的帮助开发者解决问题。

在任意位置添加：

ro.build.author=输入开发者名

ro.build.channel=输入渠道名，个人开发者统一填“OPPO”

修改版本号：

ro.build.display.id=ColorOS_机型_开发者_日期

(例如：ColorOS_I9500_CircleLee_140312)

1.5 刷机

不同机器刷机步骤略有不同，这里以三星为例：

1、将制作好的 zip 升级包 push 到 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。

2、手机先完全的关机，然后同时按住下音量下键 + HOME 键 + 电源键，等待出现英文界面

3、然后再按音量上键，进入界面为绿色机器人，此为刷机模式

4、 用 odin 刷机工具刷入第三方 recovery

5、关机后，同时按住音量上键 + HOME 键 + 电源键进入 recovery 模式

6、选择“wipe data/factory reset “和”wipe cache partition”恢复一下出厂

7、选择“install zip”

8、再选“choose zip from sdcard”

9、选择刚下载的 zip 刷机包，点确认后，选择 Yes

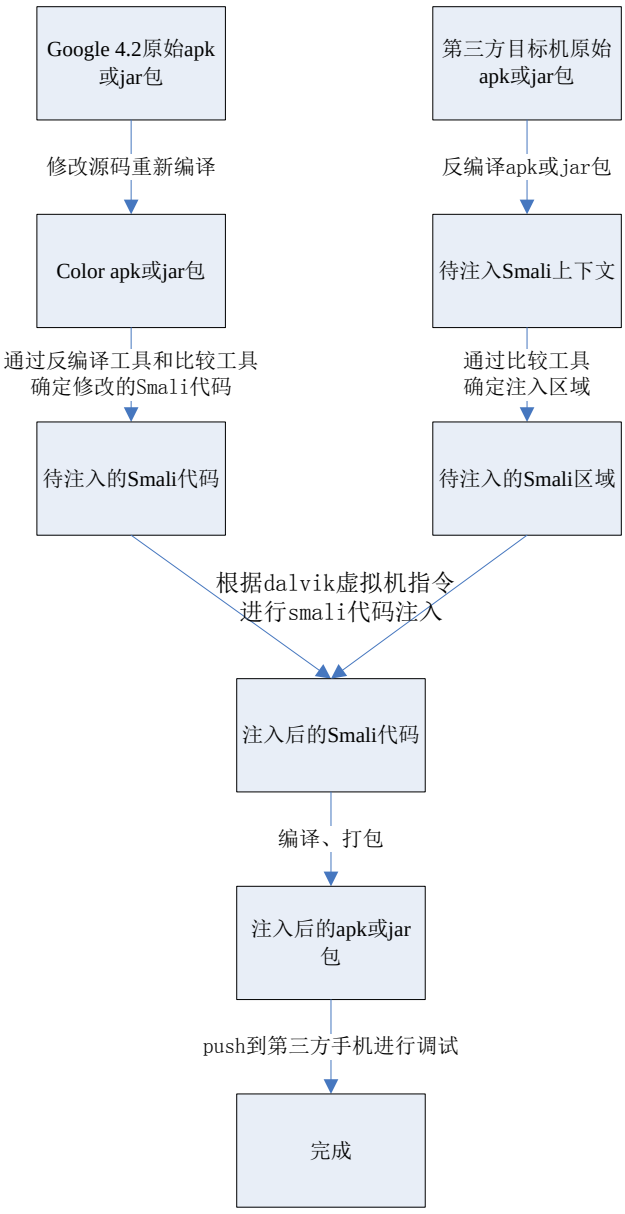
10、刷完后，返回首菜单，选择“reboot system now”重启手机。

2 移植介绍

2.1 移植详解

移植 ColorOS 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。

1、 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。

2、 编译和调试

1）使用命令可将 Smali 代码编译成 jar 包。

apktool b smali/xxx.jar.out out/framework/xxx.jar

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。

2.2 工具介绍

apktool	编译和反编译 jar 包和 apk
aapt	重新打包资源
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

2.3 移植 ColorOS Framework

在 device 路径下，执行完“make firstpatch”命令后，生成 5 个子目录：

device/update：第三方 ROM 包解压后的文件；

device/tmp_system：第三方 ROM 包解压出来的 jar 包；

device/smali：用来制作 ColorOS 订制 ROM 的 Smali 文件；

device/out:资源文件；

device/temp 路径下也有 5 个子目录：

temp/android_smali：android 源码反编译 Smali 文件（MTK 对应的是 MTK6589 源码 Smali 文件）；

temp/color_smali：ColorOS 反编译 Smali 文件；

temp/dst_smali_orig：第三方 ROM 反编译 Smali 文件；

temp/dst_smali_patched：经过脚本插桩后的第三方 ROM 的 Smali 文件；

temp/reject：ColorOS 插桩失败的 Smali 文件。

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。通过 5 大方法将 temp/reject 目录修改插桩失败文件，手动移植到 device/smali 目录下的文件。最后，执行“make fullota”，在 device 目录下生成 color-update.zip 就是生成的 ColorOS 订制升级包。

2.4 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

2.5 相机、相册问题

在适配过程中开发者可能会到这样一个问题，相机使用的是第三方自带的，而相册使用的是 ColorOS 自带的。这样就会导致相机相册无法关联起来，现象是拍照后点击照片缩略图不能跳转到 OPPO 的相册。解决方法如下：

1、使用 apktools 反编译相机的 APK 包，根据 AndroidManifest.xml 定位照片缩略图所在的 Activity 类（一般为相机启动时显示的 Activity，比如 S4 和 HTC ONE）；

2、解压相机的 APK 包，使用 dex2jar 等工具反编译其中的 classes.dex 文件，得到 jar 包，使用 JD-GUI 等工具查看 jar 包中的 java 代码，找到步骤 1 中定位的 Activity 所在的 java 类；

3、一般情况下，在该 Activity 类中可以找到响应缩略图点击事件的回调函数，跟踪并找到跳转到相册的 java 代码。同样在步骤 1 解压出来的 Smali 文件中定位该 Activity 所在的 Smali 文件，找到与跳转到相册的 java 代码对应的 Smali 代码；

4、如果 java 代码是通过包名进行跳转，则需要在 Smali 代码中把包名修改为 OPPO 相册的包名 com.oppo.gallery3d；如果 java 代码是通过特定 Action 进行跳转，则需要把 Action 修改为 com.android.camera.action.REVIEW；

5、修改后，使用 apktools 重新打包为 APK，并且进行签名。

3 注意事项

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、ColorOS 新增的 Smali 文件可以直接复制过去；

5、Phone.apk 具有平台区分性：MTK 平台和非 MTK 平台。ColorOS 会提供两套不同的 Phone.apk，MTK 平台的 Phone 适用于 MTK6589、6582、6572 等平台，非 MTK 平台 Phone 适用于高通、英伟达、展讯等平台；

6、Bluetooth.apk，Stk.apk，相机，FusedLocation.apk 这四个 apk 需使用第三方自带

apk。这里需特别注意 FusedLocation.apk 必须保留在刷机包中，否则将无法正常开机；

7、处理完 rej 文件后，打包升级。通常情况下是无法正常开机，这时候需要通过 adb logcat 打印 log 进行调试。这里有个小技巧，可以先保留 Settings.apk、SettingsProvider.apk、SystemUI.apk、OppoLauncher.apk、OppoLauncherSystem.apk、FusedLocation.apk 这 6 个 apk，其它 apk 全部删除。等能够正常开机后再导入其它 apk，这样做可以排除其它异常干扰，方便调试；

8、OppoDemos.apk 可以用来测试系统控件。如果点击出现异常，说明 Smali 移植有问题，找到报错文件进行修改；

9、如果遇到以 dalvik 开头，后面跟着 VFY 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。

10、对于一些第三方机器由于 system 分区相对较小，比如米 2。会导致 ColorOS 订制 ROM 包刷机失败。解决方法，可以将一些较大的 apk 放到 data/app 下以减少对 system 分区空间的占用，建议优先放入小欧助手（IFlySpeechService.apk，OppoSpeechService.apk，OppoSpeechAssist.apk）；