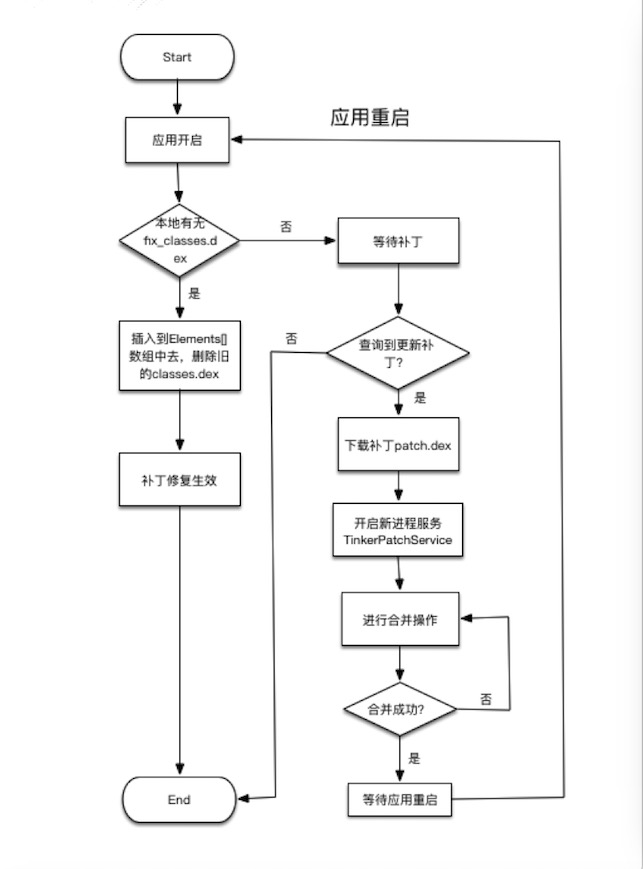
Android 热修复详细资料

1. 启动流程

Android热修复SDK首先检查是否有更新包，如果没有本地的更新包，SDK会下载补丁包，下载到补丁包之后，开启一个新的进程，合并补丁包，合并成功之后，等待应用重启，重启之后应用运行最新的程序代码，实现在线热修复。

App运行流程图



2.技术原理：

热修复方案基于android dex分包方案。一个ClassLoader可以包含多个dex文件，每个dex文件是一个Element，多个dex文件排列成一个有序的数组dexElements，当找类的时候，会按顺序遍历dex文件，然后从当前遍历的dex文件中找类，如果找类则返回，如果找不到从下一个dex文件继续查找。

理论上，如果在不同的dex中有相同的类存在，那么会优先选择排在前面的dex文件的类。 在此基础上，就是可以实现的热补丁的方案，把有问题的类打包到一个dex（patch.dex）中去，然后把这个dex插入到Elements的最前面。

替换类步骤是，构造一个ClassLoader对象，这个ClassLoader是一个继承于PathClassLoader的类，把补丁包的 Apk 路径作为参数来构造ClassLoader对象，之后通过反射替换掉 LoadedApk 的 ClassLoader。

替换so文件， ClassLoader 加载 library 的时候，将新的 library 加到数组前面，保证先加载的是新的 library。

想要更新资源文件，只需要更新Resource中的

替换资源文件， AssetManager提供了一个方法addAssetPath。在不同的 configuration 下，会对应不同的 Resource 对象，所以通过 ResourceManager 拿到所有的 configuration 对应的 resource 然后替换其 assetManager，可以实现资源文件替换。

4.主流热修复

1）阿里巴巴 [AndFix](https://github.com/alibaba/AndFix)

采用native hook的方式，这套方案直接使用dalvik\_replaceMethod替换class中方法的实现。由于它并没有整体替换class, 而field在class中的相对地址在class加载时已确定，所以AndFix无法支持新增或者删除filed的情况(通过替换init与clinit只可以修改field的数值)。 Andfix可以支持的补丁场景相对有限，另一方面，使用native替换将会面临比较复杂的兼容性问题。

2）QZone热修复方案

该方案并没有开源，但在github上的[Nuwa](https://github.com/jasonross/Nuwa)采用了相同的方式。这个方案使用classloader的方式，能实现更加友好的类替换。而且这与加载Multidex的做法相似，能基本保证稳定性与兼容性

本方案为了解决unexpected DEX problem异常而采用插桩的方式，从而规避问题的出现。事实上，Android系统的这些检查规则是非常有意义的，这会导致QZone方案在Dalvik与Art都会产生一些问题。

Dalvik； 在dexopt过程，若class verify通过会写入pre-verify标志，在经过optimize之后再写入odex文件。这里的optimize主要包括inline以及quick指令优化等。

若采用插桩导致所有类都非preverify，这导致verify与optimize操作会在加载类时触发。这会有一定的性能损耗平均每个类verify+optimize(跟类的大小有关系)的耗时并不长，而且这个耗时每个类只有一次。但由于启动时会加载大量的类，在这个情况影响还是比较大的。

Art； Art采用了新的方式，插桩对代码的执行效率并没有什么影响。但是若补丁中的类出现修改类变量或者方法，可能会导致出现内存地址错乱的问题。为了解决这个问题我们需要将修改了变量、方法以及接口的类的父类以及调用这个类的所有类都加入到补丁包中。这可能会带来补丁包大小的急剧增加。

这一套方案目前的应用成功率也是最高的，但在补丁包大小与性能损耗上有一定的局限性。特别是无论我们是否真正应用补丁，都会因为插桩导致对程序运行时的性能产生影响

3）微信热修复方案

在Qzone方案基础之上使用Bsdiff算法生补丁包，在运行时，将差异patch.dex重新跟原始安装包的旧Dex还原为新的Dex。这个过程可能比较耗费时间与内存，所以是单独放在一个后台进程:patch中。为了补丁包尽量的小，使用DexDiff算法，它深度利用Dex的格式来减少差异的大小。它的粒度是Dex格式的每一项，可以充分利用原本Dex的信息，而BsDiff的粒度是文件，AndFix/QZone的粒度为class。微信热修复方案好处：覆盖机型多，同时对补丁包大小有优化。

4.吆喝热修复技术参考微信热修复和Qzone（nuwa）热修复方案，基于classloader可以对类和资源文件的修复，覆盖机型广泛，支持多渠道打包，不同渠道分发补丁包，借鉴腾讯的差异补丁包技术，降低补丁包大小，同时对补丁包做了加密处理，通过管理后台，能够添加子账号，添加应用，上传补丁，添加渠道黑名单，机型黑名单控制补丁流量分发。

3.SDK功能简介

|  |
| --- |
| 热更新androidstudio插件用于编译补丁程序。 |
| 热更新注解，用于辅助生成application源文件。 |
| 提供热更新SDK，用于集成在现有app系统中。在APP启动时刻执行热更新补丁程序。 |
| 提供补丁管理SDK，主要用于集成在app系统中，负责下载补丁版本。 |
| 搭建补丁管理后台，可以创建账号、登陆。上传补丁包，添加机型黑名单等等。 |
| 开发过程中帮助工程师测试新补丁的调试工具。 |

5.部署流程：

1）部署mysql（可以替换oracle），tomcat。导入数据库表，部署管理程序。

2）配置androidstudio gradle插件用于编译生成补丁包，基准包。

3）配置项目注解和添加项目依赖热更新sdk，下载sdk。

4）集成appkey和appsecret，项目中配置补丁下载地址。

6.梆梆加固

已经通过梆梆安全免费版热更新验证。

7.补丁下发安全处理：

1.采用https安全协议传输补丁包程序。

2.服务器对补丁包文件再进行加密，客户端执行补丁程序之前解密。

注意事项：

这里是否支持加固，需要加固厂商明确以下两点：

1. 不能提前导入类；

2）在art平台若要编译oat文件，需要将内联取消。

3.用户基础包与补丁包混用两种模式导致补丁过大，所以编译时禁用instant run，可以在设置中禁用instant run或使用assemble方式编译。

日常debug时若想开启instant run功能，可以将热修复暂时关闭：

ext {

//for some reason, you may want to ignore tinkerBuild, such as instant run debug build?

tinkerEnabled = false

}

4.每个可能用到热修复发布补丁的版本，需要在编译后保存以下几个文件：

1）编译后生成的apk文件，即用来编译补丁的基础版本；

2）若使用proguard混淆，需要保持mapping.txt文件；

3）需要保留编译时的R.txt文件；

4）若你同时使用了资源混淆组件 AndResGuard，你也需要将混淆资源的resource\_mapping.txt保留下来，同时将r/\*也添加到res pattern中。