Android最大方法数和解决方案

目录

[1 什么是64K限制和LinearAlloc限制 2](#_Toc459974300)

[1.1 64K限制 2](#_Toc459974301)

[1.2 LinearAlloc限制 3](#_Toc459974302)

[2 谷歌分包方案 3](#_Toc459974303)

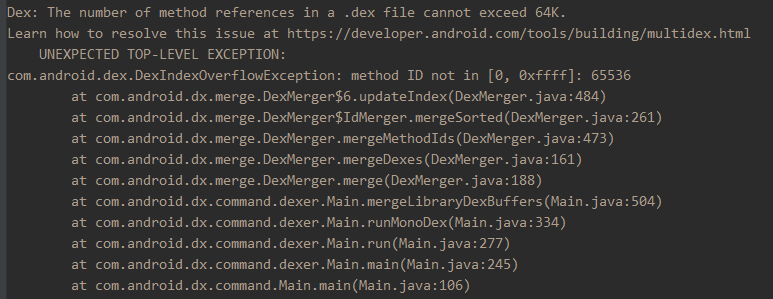
[3 谷歌multiDex存在的问题 4](#_Toc459974304)

[4 如何解决谷歌分包方案的问题 4](#_Toc459974305)

# 什么是64K限制和LinearAlloc限制

## 64K限制

随着Android应用功能的增加，代码量不断地增大，当应用方法数量超过了65536的时候，编译的时候便会提示：



这个Android著名的Dex 64k method数量上限。那么，是什么原因导致方法数不能超过64K呢？网上搜集了一下资料，原因一般有：

* **DexOpt优化的限制**：当Android系统启动一个应用的时候，有一步是对Dex进行优化，这个过程有一个专门的工具来处理，叫DexOpt。DexOpt的执行过程是在第一次加载Dex文件的时候执行的。这个过程会生成一个ODEX文件，即Optimised Dex。执行ODex的效率会比直接执行Dex文件的效率要高很多。但是在早期的Android系统中，DexOpt有一个问题，也就是这篇文章想要说明并解决的问题。DexOpt会把每一个类的方法id检索起来，存在一个链表结构里面。但是这个链表的长度是用一个short类型来保存的，导致了方法id的数目不能够超过65536个。当一个项目足够大的时候，显然这个方法数的上限是不够的。尽管在新版本的Android系统中，DexOpt修复了这个问题，但是我们仍然需要对老系统做兼容
* **dalvik bytecode的限制**：因为 Dalvik 的 invoke-kind 指令集中，method reference index 只留了 16 bits，最多能引用 65535 个方法，参考链接：<http://stackoverflow.com/questions/21490382/does-the-android-art-runtime-have-the-same-method-limit-limitations-as-dalvik/21492160#21492160>，<http://source.android.com/devices/tech/dalvik/dalvik-bytecode.html>，

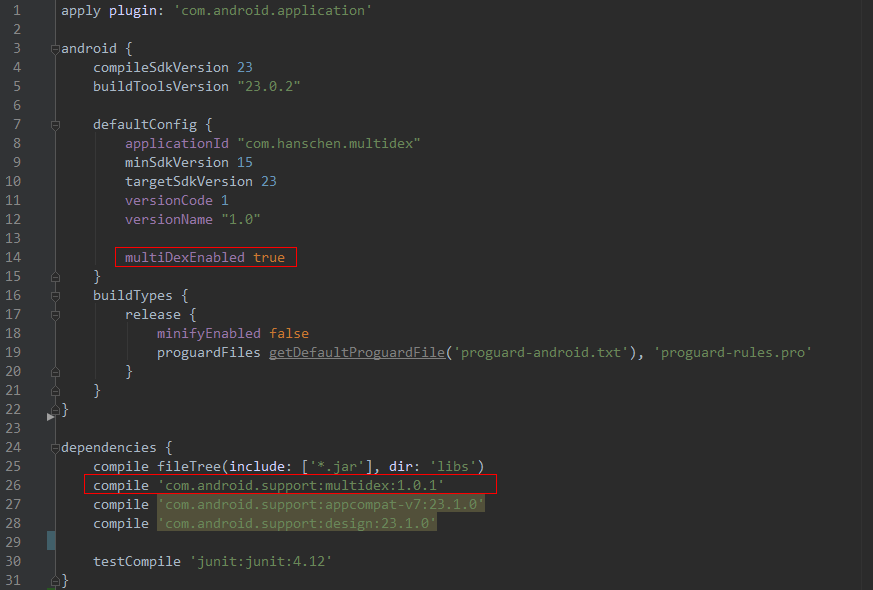
鉴于以上原因，在打包Android应用的时候，会对方法数做一个检测，当方法数超过了DexFormat.MAX\_MEMBER\_IDX（定义为0Xffff, 注意，这个不是Dex文件格式的限制，Dex文件中存储方法ID用的并不是short类型，无论最新的DexFile.h新定义的u4是uint32\_t，还是老版本DexFile引用的vm/Common.h里定义的u4是uint32或者unsigned int，都不是short类型，特此说明）便报错。

## LinearAlloc限制

即使方法数没有超过65536，能正常编译打包成apk，在安装的时候，也有可能会提示INSTALL\_FAILED\_DEXOPT而导致安装失败，这个一般就是因为LinearAlloc的限制导致的。这个主要是因为Dexopt 使用 LinearAlloc 来存储应用的方法信息。Dalvik LinearAlloc 是一个固定大小的缓冲区。在Android 版本的历史上，LinearAlloc 分别经历了4M/5M/8M/16M限制。Android 2.2和2.3的缓冲区只有5MB，Android 4.x提高到了8MB 或16MB。当方法数量过多导致超出缓冲区大小时，也会造成dexopt崩溃。

# 谷歌分包方案

谷歌提供了一个multiDex的分包方案，当方法数超过65536的时候，生成多个dex文件，把应用启动时必须用到的类和该类的直接引用类放到main dex中，把其他类放到second dex中。当应用启动之后，动态加载second dex，从而避免64k问题。使用Android Studio很容易实现分包方案：



1. 在build.gradle中添加：multiDexEnabled true
2. 加入依赖‘compile 'com.android.support:multidex:1.0.1'’
3. 让应用的Application类直接使用或者继承MultiDexApplication
4. 如果你想使用自定义的Application，又不想继承MultiDexApplication，那么可以在attachBaseContext方法里执行MultiDex.install(base)

以上就是谷歌multiDex方案所需做的设置，通过配置multiDex，便可解决64k方法数限制

# 谷歌multiDex存在的问题

虽然谷歌的分包方案很简单，但是效果并不是那么好，谷歌本身也枚举了分包方案的缺点：

1. 如果在主线程中执行MultiDex.install，加载second dex，因为加载从dex是同步的，会阻塞线程，second dex太大的话，有可能导致ANR
2. API Level 14之前，由于Dalvik LinearAlloc bug（问题22586，就是上文提到的LinearAlloc问题），multidex很可能是无法运行
3. 应用程序使用了multiedex配置的，会造成使用比较大的内存
4. 对于应用程序比较复杂的，存在较多的library的项目。multidex可能会造成不同依赖项目间的dex文件函数相互调用，找不到方法

# 如何解决谷歌分包方案的问题

针对上面的问题，参考网上的一些解决方案，如美团、facebook、微信等，初步使用的解决方法如下：

1. 第一次启动的时候，检测到未曾加载过second dex，那么启动欢迎页面（启动新的进程，原来进程进入阻塞等待，注意，此时不会发生ANR，因为已经不是前台进程了），在欢迎页面里面进行second dex的加载，加载完成后通知主线程继续
2. 设定单个dex文件最大方法数为48000（经验值）而不是65536，避免内存问题
3. 同上
4. 控制程序逻辑，未曾加载完second dex之前，进入阻塞等待，直到加载完程序才往下走

下面是流程图：

