android 混淆配置指南英文版 (https://github.com/inferjay/AndroidProguardGuide/)
我只是个默默翻译的搬运工，好吧，翻译是为了提高英文水平。

Android Proguard 指南

Android Proguard 混淆配置指南

ProGuard

这个ProGuard工具可以通过删除未使用的代码和重命名类、字段和方法与语义模糊的名字来 收缩、优化和混淆你的代码。这个结果导致你生成一个更小型号的 .apk 文件，这样会使别人反向逆推工程更加困难。 因为 ProGuard 让你的应用程序的还原工程更加困难。这个对你而言是十分有用的，当你利用这个应用特点来帮助当你的程序度过一些安全和敏感的检测当你正对你的程序授权时。

ProGuard是集成到Android 开发系统中的，因此你不必手动调用这个应用。ProGuard 只有在你编译运行程序的时候才运作，因此当你在调试模式下编译你的程序的时候，你不需要处理混淆代码。是否有 ProGuard 运行是可以完全由你有选择的，但强烈推荐你去运行它。

这个文档描述了如何去启用 和配置 ProGuard 以及使用追朔工具追踪堆栈的混淆过程和解码混淆的时候的堆栈的变化。

Enabling ProGuard

当你创建一个Android工程的时候，一个 proguard.cfg 的文件会自动的生成在项目的根目录下面。这个文件定义了 ProGuard 如何去优化和混淆你的代码，因此这是对你而言是十分重要的明白如何去定作什么是你需要的。在默认配置文件覆盖的情况下，你很有可能需要编辑什么是你自己需要的。看到下面部分关于配置 ProGuard 的信息在自定义 ProGuard 配置文件。

为了让ProGuard 能够作为 Ant 或者 Eclipse 运行构建的一部分。设置 proguard.config 的属性 在 project_root /project.properties 文件中。这个路径是可以改变的，你可以使用绝对路径或者相对路径在根目录下。

Note：当你正使用Android Studio，你需要 添加 Proguard 到你的 gradle.build 文件的构建类型。需要更多的信息，请点击Gradle Plugin User Guide (https://github.com/inferjay/GradlePluginUserGuideCN)

如果你让你的 proguard.cfg 文件离开默认的位置（项目的根目录），你可以指定它的位置好像：

```
proguard.config=proguard.cfg
```

你可以移动这个文件到任何你想要的地方，并且制定绝对的路径给它：

```
proguard.config=/path/to/proguard.cfg
```

当你构建你的应用在发布模式下，通过臀形 ant release 或者 使用 Eclipse 的Export Wizard ，这个时候构建系统会自动的检查和判断 proguard.config 的选择状态。如果开启混淆，ProGuard 会自动地处理应用的 字节代码 在打包任何东西在生成 .apk 文件之前。在调试的模式下通常不会使用 ProGuard ，因为这个会使调试的过程变得更加的繁琐。

ProGuard 在运行后会输出下面的文件：

dump.txt

描述 所有类中 在apk文件中的内部结构。

mapping.txt

列出原始和混淆过后的类，方法和字段名称。这个文件是重要的，当你从发布构建中获取一份错误报告的时候。 因为它意味着可以跟踪混淆过的堆栈耿总回到原来的类，方法、和成员的名称。由更加的信息，请参考解码混淆过的堆栈跟踪。

seeds.txt

列出没有被混淆的类和成员。

usage.txt

列出被 .apk中被剥夺的代码，这些文件在下面的目录中：

- project_root/bin/proguard 如果使用的是Ant
- project_root/proguard 如果使用的是 Eclipse

> Caution: 每一次都是在你发布的模式中运行构建。这些文件会被ProGuard最后生成的文件覆盖。每次保存一个副本当你发布你的应用程序以便释放构建反混淆错误的报告。关于更多的信息为什么要保存这些文件，看看程序发布调试的注意事项。

配置 ProGuard

在某些情况下，默认配置 的 proguard.cfg 文件就已经足够了。尽管如此，更多的情况是ProGuard 很难的去分析 和它有可能移除代码是它认为这个是没用的，但你的程序确实需要的，例如：

- 一个类 只在 AndroidManifest.xml 文件 中被引用
- 一个使用 JNI 的方法
- 动态引用的字段和方法

默认的 proguard.cfg 文件 试图覆盖的一般情况，但是你可能会遭遇到异常的情况，例如 ClassNotFoundException ，这个会发生在 当 ProGuard 除去一整个类 是你的应用程序 所需要的。

你可以通过在 proguard.cfg 添加一行 -keep 去 修复 错误当 ProGuard 移除你的代码的时候，例如

-keep public class MyClass

当你使用 -keep 选项的时候，他们是有很多的选择和考虑的。因此这是个强烈推荐 你阅读 ProGuard Manual (https://stuff.mit.edu/afs/sipb/project/android/sdk/android-sdk-linux/tools/proguard/docs/index.html#manual/introduction.html) 来获取更多的信息关于时定制你的 配置文件 。Keep选项 和 Examples 部分是特别有用的。 Troubleshooting (https://stuff.mit.edu/afs/sipb/project/android/sdk/android-sdk-linux/tools/proguard/docs/index.html#manual/troubleshooting.html) 在这个手册的故障排除中介绍了其他常见的问题，你可能会遇到你的代码被剥夺的情况。

Decoding Obfuscated Stack Traces

解析混淆过的堆栈踪迹

当你的混淆过的代码输入到一个堆栈信息中，这些方法的名字是被混淆过的。这个会让调试更加困难，这个是可能的。幸运的，无论何时 ProGuard 运行，它会输入一个 project_root/bin/proguard/mapping.txt 文件。这个会显示和组织 你的类和方法、文件名字 映射到他们混淆后的名字。

这个 windows 脚本 retrace.bat 或者 Linux Mac 的 retrace.sh 脚本 可以 让转换混淆的堆栈信息可读。它位于 sdk_root/tools/proguard/ 目录中。执行回溯工具的语言是：

retrace.bat retrace.sh [-verbose] mapping.txt []

例如：

retrace.bat -verbose mapping.txt obfuscated_trace.txt

如果你不指定一个数值，retrace 工具会从标准输入中读取。

Debugging considerations for published applications

保存 mapping.txt 当每个版本发布给用户的时候。通过保留每个副本的mapping.txt文件，可以确保你可以调试如果用户遇到的错误和用户提交混淆后的代码踪迹。一个项目mapping.txt 文件是覆盖到每一次构建项目的时候，因此你需要 关心保存哪个版本是你需要的。

例如，假设你发布一个应用程序和继续发展新功能应用程序的新版本。你可以在不久后发布构建使用 ProGuard 。这个构建 生成 的mapping.txt 会覆盖之前的 mapping.txt 文件。用户提交一个bug 报告 是在应用程序中新出版的，里面包含了一个堆栈跟踪的信息。你不再有办法调试用户的堆栈跟踪信息，因为 mapping.txt 文件与版本相关的信息是在用户的设备上面，还有其他的情况是有可能你的 mapping.txt文件 是可能被覆盖的，所以请确保你保存每一个副本文件，预测你可能需要调试的。

如何保存 mapping.txt 文件是你的决定。例如，你可以 重命名这个文件的版本号和构建的编号，或者你可以连通你的源代码进行版本控制。