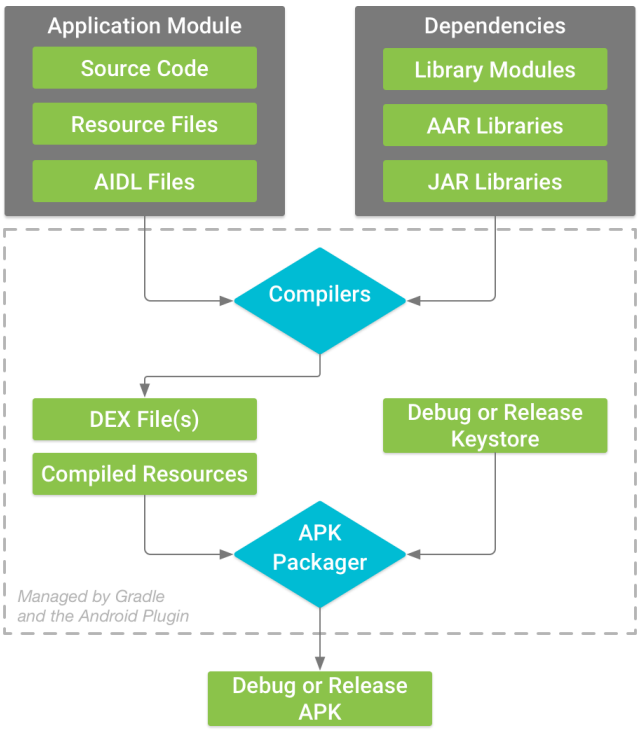
> 关于编译，你需要了解什么？https://time.geekbang.org/column/article/82468?utm_source=related_read&utm_medium=article&utm_term=related_read

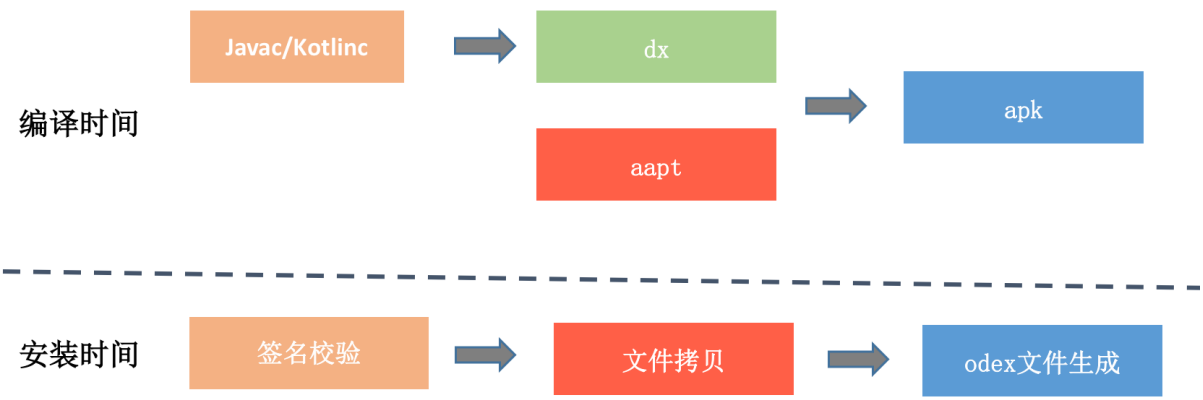
为什么优化

编译速度慢，特别项目大了，在代码开发中，增量编译十分慢。

构建流程图



优化方向



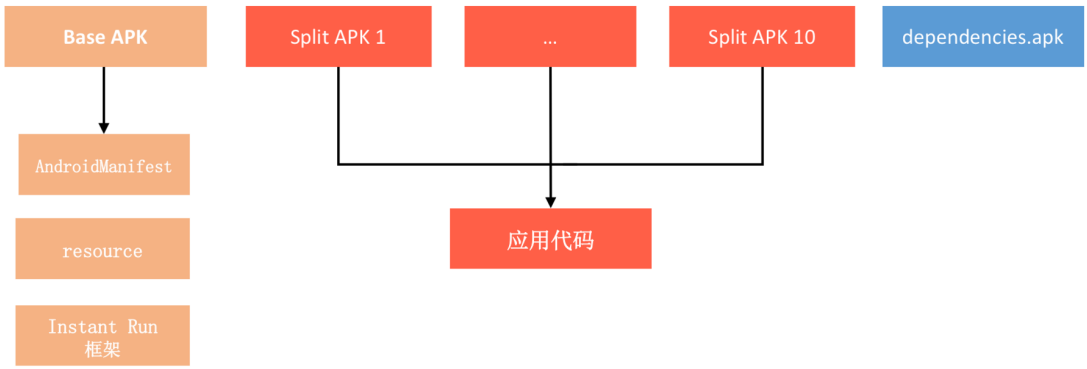
- 编译时间。把 Java 或者 Kotlin 代码编译为“.class“文件，然后通过 dx 编译为 Dex 文件。对于增量

编译，我们希望编译尽可能少的代码和资源，最理想情况是只编译变化的部分。但是由于代码之间的依赖，大部分情况这并不可行。这个时候我们只能退而求其次，希望编译更少的模块。Android Plugin 3.0使用 Implementation 代替 Compile，正是为了优化依赖关系。

- 安装时间。我们要先经过签名校验，校验成功后会有一大堆的文件拷贝工作，例如 APK 文件、Library 文件、Dex 文件等。之后我们还需要编译 Odex 文件，这个过程特别是在 Android 5.0 和 6.0 会非常耗时。对于增量编译，最好的优化是直接应用新的代码，无需重新安装新的 APK。

Instant Run

资源和 Manifest 都放在 Base APK 中， 在 Base APK 中代码只有 Instant Run 框架，应用的本身的代码都在 Split APK 中。



缺陷：

- 多进程问题。“The app was restarted since it uses multiple processes”，如果应用存在多进程，热交换和温交换都不能生效。因为大部分应用都会存在多进程的情况，Instant Run 的速度也就大打折扣
- Split APK 安装问题。虽然 Split APK 的安装不会生成 Odex 文件，但是这里依然会有签名校验和文件拷贝（APK 安装的乒乓机制）。这个时间需要几秒到几十秒，是不能接受的。
- javac 问题。在 Gradle 4.6 之前，如果项目中运用了 Annotation Processor。那不好意思，本次修改以及它依赖的模块都需要全量 javac，而这个过程是非常慢的，可能会需要几十秒。这个问题直到Gradle 4.7才解决
- 常量变更，会需要全量编译

FreeLine

比 Instant Run 更快，那是因为牺牲了正确性。也就是说它们为了追求更快的速度，直接忽略了 Annotation 和常量改变可能带来错误的编译产物

代码优化

ProGuard

主要有混淆、裁剪、优化这三大功能，耗时长

d8

Android Studio 3.0 推出了d8，并在 3.1 正式成为默认工具。它的作用是将“.class”文件编译为 Dex 文件，取代之前的 dx 工具。

除了更快的编译速度之外，还有一个优化是减少生成的 Dex 大小

R8

R8 把“.class”文件变成 Dex。还支持 ProGuard 中混淆、裁剪、优化这三大功能