iOS编译过程

1. 写入辅助文件：将项目的文件结构对应表、将要执行的脚本、项目依赖库的文件结构对应表写成文件，方便后面使用；并且创建一个 .app 包，后面编译后的文件都会被放入包中；
2. 运行预设脚本：Cocoapods 会预设一些脚本，当然你也可以自己预设一些脚本来运行。这些脚本都在 Build Phases 中可以看到；
3. 编译文件：针对每一个文件进行编译，生成可执行文件 Mach-O，这过程 LLVM 的完整流程，前端、优化器、后端；

   3.1. 预处理

   - import 头文件替换
   - macro 宏展开
   - 处理其他的预编译指令

   3.2. Lexical Analysis - 词法分析（输出token流）
   3.3.Semantic Analysis - 语法分析（输出(AST)抽象语法树）
   3.3.1 静态分析
   3.4. CodeGen - （Intermediate Representation，简称IR）IR中间代码生成
   3.5. Optimize - 优化IR
   3.6. 生成Target相关汇编 （汇编代码）
   3.7. Link生成Executable （LLVM 将会把这些汇编代码输出成二进制的可执行文件）
4. 链接文件：将项目中的多个可执行文件合并成一个mach-o文件；（内置链接器lld）

   4.1 去项目文件里查找目标代码文件里没有定义的变量。
   4.2 扫描项目中的不同文件，将所有符号定义和引用地址收集起来，并放到全局符号表中。
   4.3 计算合并后长度及位置，生成同类型的段进行合并，建立绑定。
   4.4 对项目中不同文件里的变量进行地址重定位。
5. 拷贝资源文件：将项目中的资源文件拷贝到目标包；
6. 编译 storyboard 文件：storyboard 文件也是会被编译的；
7. 链接 storyboard 文件：将编译后的 storyboard 文件链接成一个文件；
8. 编译 Asset 文件：我们的图片如果使用 Assets.xcassets 来管理图片，那么这些图片将会被编译成机器码，除了 icon 和 launchImage；
9. 运行 Cocoapods 脚本：将在编译项目之前已经编译好的依赖库和相关资源拷贝到包中。
10. 生成 .app 包
11. 将 Swift 标准库拷贝到包中
12. 对包进行签名

13. 完成打包