2019.12.18

开会笔记（cocoapods）

1. mirror update source，现在不用了
2. ~/.cocoapods是cocoapods的podspec本地缓存库。清除缓存命令为：

```
rm -rf "${HOME}/Library/Caches/CocoaPods"
```

3. podspec.json：以前是.podspec。现在是.podspec.json，可读性更高，官方也推荐json写法
4. 升级后找不到头文件（AFNetworking)，初步判断是header search path中缺失路径
5. 为了实现组件化，将三方库封装成自己的组件（.framework,.a）
6. source（源）=> 文件服务器url
7. 三方库静态source：'http://gitlab.mogujie.org/ios-team/3rdpartyframeworkspec.git'
8. 二方库静态source：'http://gitlab.mogujie.org/ios-team/mgjpodspec-framework.git'
9. 部分三方库源码source："http://gitlab.mogujie.org/CocoapodsRepos/CocoapodsRepoSpecs.git"
10. 部分二方库源码source：'http://gitlab.mogujie.org/ios-team/mgjpodspecs.git'
11. mirror source：'http://gitlab.mogujie.org/ios-team/cocoapods-official-specs-mirror.git'（目前注释不用了）
12. 组件个数达到300，打包速度就会变慢，因此添加加速source——nodependence。利用nodependence去除一些库的依赖关系，置空dependency字段，然后将dependency字段添加到nodependence source中。
13. 12的深入理解：在多重依赖时，比如A依赖B，B依赖C，在找头文件时，就会出现递归查找依赖，非常耗时间
14. nodependence source：'http://gitlab.mogujie.org/ios-team/mgjpodspec-nodependence.git'
15. source优先级是从上至下的，后面的优先级更高：第一个source需要1.0.0版本的A，第二个source需要1.2.0，那就下载第二个source1.2.0版本的A，以此类推，最后的为准。但如果是一样的版本，就下载第一个source。
16. pod update 部分依赖的三方库同一版本的warning，需要解决。就是一些组件依赖了同样版本的组件或三方库（todo）
17. rc，打tag是什么意思：

> rc::Release Candidate
> 打tag::指给git提交的某个版本打上标签，以示重要。

17. post_install代码解决三方包多文件夹嵌套：由于配置原因AFNetworking文件夹中嵌套一个或多个AFNetworking文件夹，最后一个中才是内容，导致xcode找不到包并报错，所以需要屏蔽这个bug，代码如下：

```
cd $pod
    inner=`ls |grep $pod`
    if [ \"$pod\" == \"$inner\" ]
    then
        mv $inner/* ./
        rm -rf $inner
        for file in `ls -1|grep .h` do
           followPath=`ls -l \"$file\" | awk -F ' ../' '{print
$2}'`
           ln -fs \"$followPath\" \"$file\"
        done
    fi
    cd $dir
    done
```

18. post_install中还有一些代码，目前还不理解
19. source写的不好会导致更新官方库=>变慢

> 比如组件的podfile格式不规范，添加了多余的官方source，会导致下载更新官方库所以update速度很慢。

20. xcode打包时build会两次=>真机和模拟器，所以打包速度会更慢。（需要去实验理解）
21. 高版本、低版本共存？不好管理（todo）
22. 壳工程，pods文件夹占用硬盘空间大，软连接节省空间？（todo）

一周规划

需要做的内容：cocoapods版本更新

具体方案

1. 找到头文件报错的一些三方库和没有问题的三方库，列表对比，找出区别
2. 将3rd的podspec内容fork下来搭建一个新工程，尝试pod update多个三方库，并进行第一步表格记录
3. 记录发现是字段问题还是设置问题，目前主要是以下两个字段，看看是不是有新的字段更新，或者是value值设置不正确

```
"public_header_files": "Headers/*",
"source_files": "Headers/*",
```

4. 在本地pod update时可能会出现没有效果，但不一定是真的无效，可能是podupdate从本地缓存获取而不是重新拉下来，可以执行清除本地缓存命令：

```
rm -rf "${HOME}/Library/Caches/CocoaPods"
```

> pod update的时候可以加上--help，添加一些命令，比如pod update --verbose，可以看到更多打印信息，更新的细节

5. cocoapods版本从1.5更新到1.6.0后有较大的变化，看一下更新内容，是否有影响，还可以在上面查询相关问题
6. 用屏幕共享在ip为10.50.95.14电脑上操作，zip操作header或者zip整个文件夹，看是否有这个原因。可以新建AFNetWroking2.4.1的版本进行测试操作，不会影响业务使用。然后本地repo pod update，并且将podspec中version以及source修改为2.4.1版本
7. 一周时间看看工作结果，没有进展就需要从cocoapods源码入手，每天跟进。
8. 熟悉命令行的使用

fork测试

1. 新建工程不报错？？ => hook没有加上的原因？ => 不是
2. 更换思路 => 从headers/public文件夹中分析，为什么有些可以生成，有些不行

测试方式：

1. 打开文件夹（1.8.4）mogujie-iphone/Pods/Headers/Public ，查看public成功生成的三方库
2. 打开文件夹 /Users/gemini/Desktop/iOS/podspecs/3rdpartyframeworkspec，查看3rd的所有库
3. 对比3rd中哪些库生成成功，并观察podspec的区别
4. 发现只有.a库才可以生成headers，.framework不能

2019.12.19

工作内容

1. 尝试在public_header_files和source_files添加头${PODS_ROOT}，失败
2. 尝试header_dir字段，这个不相关
3. 发现问题，source_files字段地址不对。（后面发现没什么问题）
4. 在字段路径前加上AFNetworking.framework/或AFNetworking/没效果
5. 目前来看与AFNetworking.podspec中public_header_files字段关系不大。
6. 修改ip：10.50.95.14中的zip包，将headers文件夹另外打包，可以成功生成。（这是妖路子）
7. 两个测试方案：一个是继续更改zip包格式，并在高版本mogujie工程中测试是否还报错；还一个是继续修改source_files的字段。（后面发现都没效果）
8. 结果：修改source_files没什么进展,看github的issue没什么相关的，然后1.6版本修改也没进展 10.寻找相关改动 11.module_map？（不相关） 12.继续查看改动

2019.12.20

工作内容

1. AFNetworking.podspec.json中在subspec或主spec中添加删除vendored_frameworks字段——都没有效果
2. AFNetworking.podspec.json中添加static_framework字段——无效。
3. 发现Pods中AFNetworking文件夹下缺少内容，重新拉一个试试=>未知操作导致的，后面try时需要注意这个文件夹是否被改动了=>测试发现：设置public_header_files&source_files字段如下所示，添加AFNetworking/AFNetworking.framework/路径就会出现该问题，只留下Headers文件夹以及其中一个AFNetworkReachabilityManager.h文件.=>tips:AFNetworking/——这个dir有毒。
4. 1和2两点之前应该是在3的情况下测试的，可以尝试再试试，但大概率也是不成功的，有空再尝试
5. 经过4的重新测试，发现不是这个问题.
6. 还是要找到public中不生成包的确切改动：(1)从github上cocoapods的release中搜索关键内容;(2)查看ruby源码，搜索public、ls等关键字

学长沟通

1. 新建工程不报错，可以与高版本主工程对比xcode配置，header search path，user search path，看看修改后是否也会报错。
2. 递归选项不能设置，必须保持递归选项关闭

2019.12.23

工作内容

> 对比xcode设置

1. 没有CI
2. framework search paths 删除$(SRCROOT) $(DEVELOPER_FRAMEWORKS_DIR) 无效 $(inherited) 改为recursive ，无效
3. 修改build setting中所有配置选项，没有进展

> release版本改动

1. 发现1.4版本对Header_search_path改动较多
2. 在AFNetworking.podspec.json中添加以下字段，没有效果

```
"pod_target_xcconfig" : {
    "OTHER_LDFLAGS" : "$(inherited) -framework 'AFNetworking'",
    "CLANG_ALLOW_NON_MODULAR_INCLUDES_IN_FRAMEWORK_MODULES" :
"YES",
    "FRAMEWORK_SEARCH_PATHS" : "$(inherited)
'${PODS_ROOT}/AFNetworking'"
},
```

```
"static_framework" : true,
"prefix_header_file" : false,
"pod_target_xcconfig" : {
    "HEADER_SEARCH_PATHS" : "${PODS_TARGET_SRCROOT}"
},
```

沟通

1. 除了解决问题外，还需要从中学到东西，很多业务的或是社招进来的，对coocapods类似的东西，都不懂，很多需要问到我们团队，需要很快给出解决方案，学长就有很多的经验和踩过这些类似的坑！
2. 延迟享受，在工作最初的几年不要去享受消费，要做正确的工作选择、消费选择。
3. 买房大于买车，可能刚毕业都说不买房，到最后还是决定买房，买车是先快乐，然后痛苦，买房是买的时候痛苦，后面都是快乐的，房贷是给自己的动力，更加努力地工作。
4. 年前会做明年工作规划
5. 后面可能会做GPU Image优化，或者AI在客户端的工程搭建等，都会是我们团队的工作。
6. 有些问题解决不了是正常的，但是肯定是有解决方案的，可以找厉害的人解决，最重要的是在这个过程学习到东西，记住踩过的坑。

2019.12.24

工作内容

1. podfile修改：添加如下代码，没有效果

```
target 'Mogujie4iPhone' do
    inherit! :search_paths
end
```

2. podfile中注释以下代码，没有效果。应该是修改某个库的build_settings

```
if target.name == "Pods-Mogujie4iPhone Today Extension-
AFNetworking"
    target.build_configurations.each do |config|
        config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||=
['$(inherited)', 'AF_APP_EXTENSIONS=1']
    end
end
```

3. 发现Pods/AFNetworking/Reachablity中没有生成AFNetworkReachabilityManager.h文件
4. cocoapackge打包替换之前的命令生成方式，观察framework目录架构是否变化

命令方式

1. 命令一：xcodebuild -workspace "AFNetworking.xcworkspace" -scheme "AFNetworking iOS" -configuration Release -sdk iphoneos -derivedDataPath ./build clean build 2>&1
2. 命令二：xcodebuild -workspace "AFNetworking.xcworkspace" -scheme "AFNetworking iOS" -configuration Release -destination 'platform=iphonesimulator,name=iPhone 8' -derivedDataPath ./buildsimu clean build
3. 命令三：lipo -create /Users/gemini/Desktop/iOS/AFNetworking/build/Build/Products/Release-iphoneos/AFNetworking.framework/AFNetworking /Users/gemini/Desktop/iOS/AFNetworking/buildsimu/Build/Products/Release-iphonesimulator/AFNetworking.framework/AFNetworking -output AFNetworking
4. 生成AFNetworking文件后替换/Release-iphoneos/AFNetworking.framework/AFNetworking 即可与Headers等文件打包成AFNetworking.framework.zip到服务器中提供下载

cocoapods-package

1. 使用cocoapods-package 打包AFNetworking,然后在服务器中新建2.5.2.1版本，并尝试该版本
2. 出现403forbidden，因为直接把文件拉过去没有设置好文件权限，需要设置为所有人可读

2019.12.25

工作内容

1. 源码中查找public_header_path相关源码
2. framework路径等相关源码在prepare_artifacts.script.rb
3. 将1.8.1版本cocoapods与1.2.1版本源码进行对比，查看改动在哪里
4. Add install! 'cocoapods', :disable_input_output_paths => true to the Podfile.（这条改动用来阻止在[cp]copy resource paths中生成input output files）
5. xcconfig完全不同，1.2.1和1.8.4版本

2019.12.26

工作内容

重大突破

1. ~~成功找到报错原因，在pods/target/MGJHangDetector中没有配置framework路径，即AFNetworking路径，加上就成功了，可以用post_install实现，代码如下~~更新，是MGJHangDetector没有添加对MGJCategories的依赖！

```
if target.name == "MGJHangDetector"
    target.build_configurations.each do |config|
        config.build_settings['FRAMEWORK_SEARCH_PATHS'] ||=
['$(inherited)', '${PODS_ROOT}/AFNetworking']
    end
end
```

2. 但是手动去修改不现实，最好还是做到把publiic目录加回来，同时添加header_search_path
3. 因为把文件目录改了就可以，所以看看源码中搜索路径有什么不同
4. 现在的情况是public目录中没生成以及在header_search_path没有.framework的public头文件路径，两者必须都要
5. installer/xcode/pods_project_generator/pod_target_installer.rb:1.8.1

2019.12.27

cocoapods源码学习笔记

1. 入口文件——installer.rb

```
###入口函数
def install!
    prepare
    resolve_dependencies
    download_dependencies
    validate_targets
    if installation_options.skip_pods_project_generation?
        show_skip_pods_project_generation_message
    else
        integrate
    end
    write_lockfiles
    perform_post_install_actions
end
```

2. 各函数功能如图

运行流程

1. installer.rb创建pods路径下所有文件： integrate->generate_pods_project->create_and_save_projects->create_generator->Xcode::MultiPodsProjectGenerator->generate!->create_pods_project->PodsProjectGenerator->install_file_references->FileReferencesInstaller->add_file_accessors_paths_to_pods_group->link_headers
2. 在以上运行过程中发现，1.2.1和1.8.1版本中link headers函数有区别。 其中1.2.1中有以下一句代码：

```
unless pod_target.requires_frameworks?
    vendored_frameworks_header_mappings(headers_sandbox,
file_accessor).each do |namespaced_path, files|
        sandbox.public_headers.add_files(namespaced_path, files)
    end
end
```

3. vendored_frameworks_header_mappings函数如下：

```
# Computes the destination sub-directory in the sandbox for headers
# from inside vendored frameworks.
#
# @param  [Pathname] headers_sandbox
#         The sandbox where the header links should be stored for
this
#         Pod.
#
# @param  [Sandbox::FileAccessor] file_accessor
#         The consumer file accessor for which the headers need to
be
#         linked.
#
    def vendored_frameworks_header_mappings(headers_sandbox,
file_accessor)
        mappings = {}
        file_accessor.vendored_frameworks.each do |framework|
            headers_dir =
Sandbox::FileAccessor.vendored_frameworks_headers_dir(framework)
            headers =
Sandbox::FileAccessor.vendored_frameworks_headers(framework)
            framework_name = framework.basename(framework.extname)
            dir = headers_sandbox + framework_name
            headers.each do |header|
                # the relative path of framework headers should be
kept,
                # not flattened like is done for most public
headers.
                relative_path =
header.relative_path_from(headers_dir)
                sub_dir = dir + relative_path.dirname
                mappings[sub_dir] ||= []
                mappings[sub_dir] << header
            end
        end
        mappings
    end
```

4. 仔细观看发现1.2.1是在以上代码另外加了framework的headers，在下面代码中利用.framework后缀屏蔽了framework的headers路径添加

```
header_mappings(headers_sandbox, file_accessor,

file_accessor.headers).each do |namespaced_path, files|
    pod_target.build_headers.add_files(namespaced_path,
files.reject { |f| f.to_path =~ framework_exp })
end

header_mappings(headers_sandbox, file_accessor,
file_accessor.public_headers).each do |namespaced_path, files|
    sandbox.public_headers.add_files(namespaced_path, files.reject
{ |f| f.to_path =~ framework_exp })
end
```

5. 在1.8.1中没有另外添加代码，只有以下两个遍历方法

```
pod_target_header_mappings =
pod_target.header_mappings_by_file_accessor.values
pod_target_header_mappings.each do |header_mappings|
    header_mappings.each do |namespaced_path, files|
        pod_target.build_headers.add_files(namespaced_path, files)
    end
end

public_header_mappings =
pod_target.public_header_mappings_by_file_accessor.values
public_header_mappings.each do |header_mappings|
    header_mappings.each do |namespaced_path, files|
        sandbox.public_headers.add_files(namespaced_path, files)
    end
end
```

6. 其中pod_target的public_header_mappings_by_file_accessor和header_mappings_by_file_accessor两个方法中就对framework做了过滤处理，函数如下,可以判断只有是non_library_specification?为false时才可以添加路径

```
valid_accessors = file_accessors.reject { |fa|
fa.spec.non_library_specification? }
```

2019.12.30

工作内容

1. 尝试修改高版本cocoapods本地源码并运行
2. 两个命令:

```
### 利用cocoapods.gemspec将自己修改过的cocoapods源码生成cocoapods.gem文件
sudo gem build cocoapods.gemspec
### 利用cocoapods.gem安装自己的cocoapods
sudo gem install --local cocoapods-1.8.4.gem -n /usr/local/bin
```

3. ~~直接在/Library/Ruby/Gems/2.6.0/gems/cocoapods-1.8.4目录下修改源码没有实际效果~~（是有效果的，只是当时改的代码没有运行到）
4. 目前判断需要生成一个新的gem包并install
5. 从github上clone cocoapods的代码，做修改后利用上面的命令生成.gem文件，然后第二个命令下载cocoapods，就可以修改源码
6. 直接在/Library/Ruby/Gems/2.6.0/gems/cocoapods-1.8.4上修改代码即可，3中没有效果是因为写的代码没运行到！！！！！！ 7.在file_references_installer中添加以下代码，可以生成public文件，但是编译非常慢且不成功。

```
pod_target.file_accessors.each do |file_accessor|
   headers_sandbox = Pathname.new(file_accessor.spec.root.name)
   vendored_frameworks_header_mappings(headers_sandbox,
file_accessor).each do |namespaced_path, files|
        sandbox.public_headers.add_files(namespaced_path, files)
    end
end
```

8. 报错如下,这个报错原因主要与pods库资源路径有关，可能是太多太长了。

```
/Users/gemini/Desktop/iOS/mogujie-iphone/Pods/Target Support
Files/Pods-Mogujie4iPhone_all-Mogujie4iPhone/Pods-
Mogujie4iPhone_all-Mogujie4iPhone-resources.sh: line 7: realpath:
command not found
:1134: error: Unexpected failure
```

2019.12.31

工作内容

1. Build Phases中[CP]Copy Pods Resources中勾选Run script only when installing就可以成功运行

> 因为勾选了在build时就会跳过run script过程，但运行起来的app没有任何资源（图片等）

2. file_references_installer中add_file_accessors_paths_to_pods_group方法有所区别，可能导致resource.sh文件变化
3. 2中没有什么发现
4. 将问题锁定在file_references_installer.rb中新添加语句上，是不是没有添加头文件索引相关的方法
5. sandbox_header_paths_installler.rb中install_target方法
6. headers_store.rb中add_search_path和add_files两个方法

1.2.1和1.8.4版本header mapping区别

1.2.1

1. 使用header_mappings方法
2. 使用vendored_frameworks_header_mappings方法，该方法中调用vendored_frameworks_headers_dir方法生成路径
3. 直接再linking headers中使用add_search_paths

1.8.4

1. 直接使用pod_target.header_mappings_by_file_accessor.values
2. header_mappings方法移到pod_target.rb中，同时还添加了以下代码，导致.framework不在headers/public下产生link headers，注释就可以生成

```
next if header.to_s.include?('.framework/')
```

3. 在file_accessor.rb中存在public_headers方法，默认include_frameworks = false，尝试设置为true

todo

1. 以下三个改动逐一测试，看看是不是必须3个都在

> file_accessor.rb->def public_headers(include_frameworks = false) -> false 改为true

> pod_target.rb->next if header.to_s.include?('.framework/')注释

> target_integrator.rb->MAX_INPUT_OUTPUT_PATHS = 1000->设置为200

2. 在工程目录下有Scripts文件夹，都是工程脚本，写一个脚本尝试修改cocoapods源码，可以写一下伪代码和学长确认是否可以，主要对路径的一个处理
3. 修改MGJ-categories的依赖，看看是不是这里的问题

2020.01.02

工作内容

1. 写好脚本后，需要测试下pod update和build的时间对比（与1.2.1对比）
2. 尝试修改MGJ-Categories.podspec.json：删除default_subspecs字段无效，修改dependencis中AFNetworking/Reachablity为AFNetworking，无效
3. 编写脚本，学习grep、awk、sed3个命令的使用，学习正则表达式的使用
4. 要怎么把脚本执行？？ sh a.sh文件手动执行
5. 下周要开始准备分享文档，cocoapods升级的说明，做了哪些工作，为什么选择这个方案
6. 除了脚本、还要找更好的方案。比如podspec、podfile上的代码修改
7. 测试几个方案的编译时间、pod update时间
8. shell代码会给到学长那里review

沟通

1. 目前给的压力不大，年后会给任务的deadline，给压力。
2. 年后的规划可能有端上AI，所以要自己了解一些模型训练的知识。
3. 年后会把一部分的工作转到我这里，比如服务迭代等
4. 空余时间对公司源码等可以学习下，做好时间管理
5. 列举了一些人空余时间学习的成功，鼓励学习，好处会在一年两年后体现
6. 对每星期的总结要更有深度，有自己的思考，这很重要。然后会主动地交流一个月的学习成果等
7. 做好未来规划
8. 下周要开始准备分享文档，cocoapods升级的说明，做了哪些工作，为什么选择这个方案

2020.01.03

工作内容

1. 测试pod update和build时间
2. 尝试修改podspec
3. **发现只需要在mgjhangdetector里面增加static_framework : true即可**
4. 蘑菇街工程只有静态库，没有动态库，动态库基本都是官方库，mgjhangdetector是源码库
5. 源码库不能依赖静态库？必须添加static_framework podspec attribute，源码库才能依赖上静态库？原文如下

> Address #6772

> CocoaPods/Core#386 adds the static_framework podspec attribute.

> This PR uses that attribute to build a source pod into a static library framework. All dependents and dependencies should interact with the source pod just as if it were a vendored_framework static library framework CocoaPod.

6. 针对以下错误，尝试安装home-brew，并安装coreutils，不报line7:realpath错误了，但还是有1134: error: Unexpected failure。

```
resources.sh: line 7: realpath: command not found
:1134: error: Unexpected failure
```

```
brew install coreutils
```

7. 注释掉那行代码，是否会对动态库造成影响
8. 需要注意后续的开发维护成本等

2020.01.06

工作内容

1. 解决unexpected failure错误,确定MAX_INPUT_OUTPUT_PATHS属性的作用，了解值修改后的影响。1000改成200会不会导致编译变慢。

> 后面发现将1000改为200的效果等同于添加disable_input_output_paths选项，不生成input_output列表

2. ~~尝试在cocoapods中提出问题~~
3. disable_input_output_paths
4. install_resource
5. MAX_INPUT_OUTPUT_PATHS指的是资源输入输出路径最大限制值,注释如下，比如.cfg/.js/.bundle等资源

```
# @return [Integer] the maximum number of input and output paths to
use for a script phase
```

6. 从resource.sh文件中的代码可以看出，使用trap获取中断，并打印是在哪一行出现报错。on_error函数可以获取报错文件绝对路径

```
function on_error {
  echo "$(realpath -mq "${0}"):$1: error: Unexpected failure"
}
trap 'on_error $LINENO' ERR
```

7. 在podfile中使用install! => disable_input_output_paths选项，可以避免在build phases的copy_中生成input_file和outpub_file的路径。这些路径如果生成会导致unexpected failure错误
8. 使用install => generate_multiple_pod_projects选项可以将pods生成xcode工程，减少整体工程大小，缩短indexing过程
9. 测试几个方案的工程编译时间
10. 总结之前看的github上的讨论以及查到的资料，编写文档说明为什么加上这几个字段可以解决高版本出现的问题！！！！！

2020.01.08

工作内容

1. 整理解决方案的实现原理（11：00前）
2. 做文档
3. https://blog.cocoapods.org/CocoaPods-1.4.0/
4. 发现编译到下面这句后，就卡很久，有时间调查原因

```
Showing All Messages
CoreGraphics PDF has logged an error. Set environment variable
"CG_PDF_VERBOSE" to learn more.
```

5. 将static_framework = true改动加到iOS-team库中
6. clone下来mgjpodspec库，并修改MGJHangDetector的podspec.json文件，添加static_framework字段

重大突破

1. 发现不需要添加static_framework字段。
2. 发现造成头文件找不到的原因如下：

> 在mgjpodspec_nodependence中存在mgjhangdetector库buff，其中podspec.json中dependencies为空。由于podfile的执行顺序是从上向下，在mgjpodspec_nodependence的source中发现优先级更高，只要照到了mgjhangdetector，就不会使用mgjpodspec中的mgjhangdetector.podspec.json了，所以依赖没有加上，导致bug

3. 明天在以上基础下继续做测试，build测试，pod update测试，分install和generating
4. 并作出更新文档初版，后面学长review后会群发邮件！
5. 深入研究build时出现的延时长的几个问题
6. TODO:测试没加generate_multiple_pod_projects的build时间
7. TODO:测试加了generate_multiple_pod_projects以及去掉disabe_input_output是否可用
8. TODO:测试加了generate_multiple_pod_projects的build时间

2020.01.09

工作内容

1. 测试不加disable_input_output_files为什么报错
2. 重新拉工程，并用了源码的shell脚本，会有头文件问题，重新拉一个再试试
3. 用源码编译工程会有bug，头文件有""<>两种引用同时存在

今日测试

1.8.4版本	test1	test2	test3	不clean或删除DerivedDate继续编译
总时长	299.4s	306.0s	281.1s	6-8s左右
precompile .pch	11.1s	10.2s	11.2s	-
linking	87.3s	82.3s	89.8s	-
run script	146.9s	155.7s	127.4s	-

1.2.1版本	test1	test2	test3	不clean或删除DerivedDate继续编译
总时长	273.0s	301.2s	326.2s	155.4s
precompile .pch	10.3s	12.3s	12.6s	-
linking	81.1s	75.4s	96.2s	-
run script	121.9s	126.7s	125.7s	134.8s

1.8.4-generate_multiple_pod_projects	test1	test2	test3	不clean或删除Derived Date继续编译
总时长	277.0s	294.2s	316.9s	157.7s
precompile .pch	10.9s	10.4s	12.3s	-
linking	88.5s	77.8s	95s	-
run script	132.0s	139.0s	141.4s	137.0s

1.8.4修改代码后build	test1	test2
总时长	86.7s	88.4s
precompile .pch	-	-
linking	78.5s	79.4s
run script	-	-

1.2.1修改代码后build	test1	test2
总时长	234.0s	238.5s
precompile .pch	-	-
linking	81.8s	92.3s
run script	140.3s	134.0s

1.8.4pod update	test1	test2
总时长	81s	67s
Analyzing dependencies	10s	8s
downloading	10s	12s
Generating Pods project	40s	38s
Integrating client project	2s	4s

1.2.1pod update	test1	test2
总时长	118.0s	144.0s
Analyzing dependencies	20s	39s
downloading	15s	23s
Generating Pods project	62s	58s
Integrating client project	9s	9s