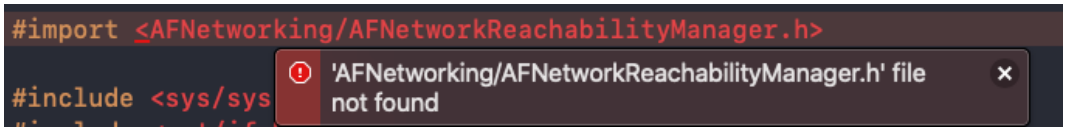
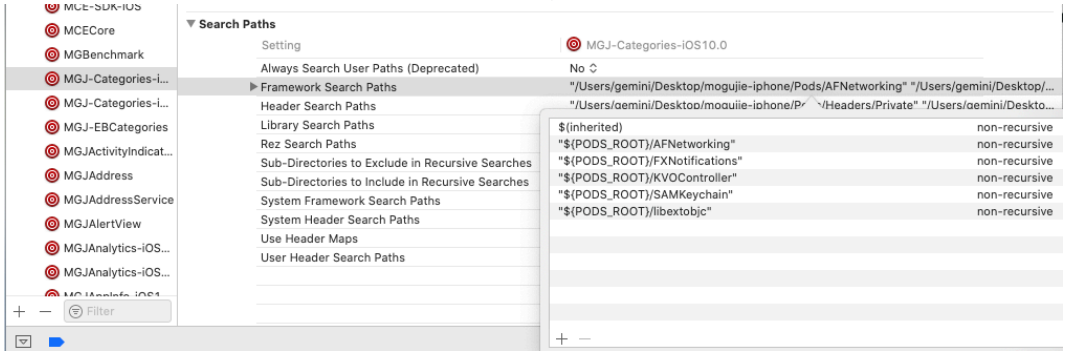
cocoapods版本升级

1. **当前版本**:1.2.1，**升级版本**:1.8.4
2. **升级后问题**:如图所示，编译不通过，在MGJ-Categories/UIDevice+MGJKit.h中报错，找不到文件。



查找报错原因查找

1. 查看MGJ-Categories/Build Settings/Search Paths，可以发现所有依赖的搜索路径都存在，这个问题不是由MGJ-Categories引起的。



2. 如图所示，由报错信息可以看出这个调用过程是从MGJHangDetector库发起的。

3. 查看MGJHangDetector/Build Settings/Search Paths，发现framework_search_paths是空的，再查看Build Phases/Dependencies，果然没有添加上MGJ-Categories的依赖。
4. 查看MGJHangDetector.podspec.json文件，dependencies字段中添加了MGJ-Categories依赖，没有任何问题。
5. 再找nodependence源，MGJHangDetector.podspec.json中的dependencies字段被置空了，成功找到问题。

解决方案

- 将mgjpodspec-nodependence源中将MGJHangDetector库移除。

原因分析

依赖未添加原因

1. podfile中source的优先级是由上至下的，相同版本的库以第一个找到该库的源为准。
2. Podfile中所有的源如下，可以看到mgjpodspec-nodependence源处在第一位。

```
# 主客加速专用source，组件工程中禁止添加这一行
source 'http://gitlab.mogujie.org/ios-team/mgjpodspec-
nodependence.git'
# 三方库framework源，注释该行即可使用源码
source 'http://gitlab.mogujie.org/ios-
team/3rdpartyframeworkspec.git'
source 'http://gitlab.mogujie.org/ios-team/mgjpodspec-
framework.git'

source 'http://gitlab.mogujie.org/ios-team/mgjpodspecs.git'
source
"http://gitlab.mogujie.org/CocoapodsRepos/CocoapodsRepoSpecs.git"
#source 'http://gitlab.mogujie.org/ios-team/cocoapods-official-
specs-mirror.git'

# IM 源
source 'http://gitlab.mogujie.org/im/IMPodSpecs.git'
```

1.2.1不报错原因

1. 1.2.1版本中，MGJHangDetector库同样没有添加MGJ-Categories依赖。
2. cocoapods1.2.1版本中，pod都会在Pods/Headers/下生成目录，而所有pod的Search Paths/Header Search Paths中都添加有"${PODS_ROOT}/Headers/Public"搜索路径，因此即使没有加上MGJ-Categories的依赖，同样能找到所有文件的位置。

升级后测试

编译测试

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

结论:编译时间基本接近，1.8.4版本在run script过程耗时较长一些。在编译完成后不做任何改动继续编译，1.8.4版本6-8s即可完成，1.2.1版本则需要155s。查看编译过程发现1.8.4版本linking和run script都利用第一次编译缓存，而1.2.1版本则只利用第一次linking的缓存，耗时都在run script上。

完成第一次编译后修改代码再编译测试

1.8.4版本	test1	test2
总时长	86.7s	88.4s
precompile .pch	-	-
linking	78.5s	79.4s
run script	-	-

1.2.1版本	test1	test2
总时长	234.0s	238.5s
precompile .pch	-	-
linking	81.8s	92.3s
run script	140.3s	134.0s

结论:1.2.1版本在第一次编译后修改代码，会重新进行linking和run script过程，总时长基本等于第一次编译。1.8.4版本在修改代码后不会重新run script，但会linking，所以编译时间缩短很多。

pod update后编译测试

- 编译时间等同于清空缓存重新编译。

pod update时间测试

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

1.2.1版本	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

结论:1.8.4版本在pod update耗时上优化效果明显。除了downloading时长与网络状况相关，Analyzing、Generating、Integrating等过程都有大幅度的耗时降低。