

xcconfig 指南

xcconfig 文件的语法比较简单，每个配置文件都由一系列键值分配组成，这些键值分配具有以下语法：

```
1 | BUILD_SETTING_NAME = value
```

注释

xcconfig 文件只有一种注释方式 `//` 。

include 导入其他设置

在创建 xcconfig 文件的时候，可以根据需求，创建多个。也就意味着，可以通过 include 关键字导入其他的 xcconfig 内的配置。通过 include 关键字后接上双引号：

```
1 | #include "Debug.xcconfig"
```

在搜索引入的文件时，如果是 `/` 开头，代表绝对路径，例如：

```
1 | // 表示确切的文件位置
2 | #include "/Users/ws/Desktop/VIP课程/第一节、符号与链接/强化班第一节课资料/完成代码/LoginApp-冲突/Pods/Target Support Files
   | /Pods-LoginApp/Pods-LoginApp.debug.xcconfig"
```

或者通过相对路径，以 `${SRCROOT}` 路径为开始：

```
1 | #include "Pods/Target Support Files/Pods-LoginApp/Pods-LoginApp.debug.xcconfig"
```

变量

变量定义，按照 `OC` 命名规则，仅由大写字母，数字和下划线（`_`）组，原则上大写，也可以不。字符串可以是 `"` 也可以是 `'` 号。
变量有三种特殊情况：

1. 在 `xcconfig` 中定义的变量与 `Build Settings` 的一致，那么会发生覆盖。可以通过 `$(inherited)`，让当前变量继承变量原有值。例如：

```
1 | OTHER_LDFLAGS = -framework SDWebImage
2 | OTHER_LDFLAGS = $(inherited) -framework AFNetworking
3 | // OTHER_LDFLAGS = -framework SDWebImage -framework AFNetworking
```

注意⚠：有部分变量不能通过 `xcconfig` 配置到 `Build Settings` 中，例如：配置 `PRODUCT_BUNDLE_IDENTIFIER` 不起作用。

2. 引用变量，`$()` 和 `${ }` 两种写法都可以：

```
1 | VALUE=Cat
2 | TEACHER=$(VALUE)-${VALUE}
```

3. 条件变量，根据 `SDK`、`Arch` 和 `Configuration` 对设置进行条件化，例如：

```
1 | // 指定`Configuration`是`Debug`
2 | // 指定`SDK`是模拟器，还有iphonios*、macosx*等
3 | // 指定生效架构为`x86_64`
4 | OTHER_LDFLAGS[config=Debug][sdk=iphonesimulator*[arch=x86_64]= $(inherited) -framework "Cat"
```

注意⚠：在 `Xcode 11.4` 及以后版本，可以使用 `default`，来指定变量为空时的默认值：

```
1 | $(BUILD_SETTING_NAME:default=value)
```

优先级（由高到低）：

1. 手动配置 Target Build Settings
2. Target 中配置的 xcconfig 文件
3. 手动配置 Project Build Settings
4. Project 中配置的 xcconfig 文件