使用

- 安装python库
  - `pip install pbxproj`
  - `pip install Cheetah`
- 配置工作目录
  - 新建一个目录,比如叫workspace。
  - 把马甲包工具copy到工作目录。
  - 建一个资源目录，比如叫resources。
  - 在资源目录下建立一个马甲包目录，比如叫proj1。
  - 把打包工具目录repack/data下配置文件复制到马甲包资源目录pro1下。并根据项目情况进行修改里面的配置文件。
  - 执行打包命令: `python repack/repack.py -s 源工程目录 -o 输出目录 -r 资源目录(指向刚才建立的resources) -d repack/data -c 工程配置(resources/Test/project.json) --step-config 执行步骤配置(resources/Test/step.json)`

说明

程序入口repack.py

- 参数说明:

参数	说明
src-project	原工程目录。
out-dir	输出目录。混淆工具输出结果。
resource-dir	资源目录。马甲包有关的资源。比如，图标，启动图，游戏内容图等。
config-file	配置文件。配置项目相关参数。比如项目名称，xcode工程位置，开发者账，以及action中会用到的参数。
step-config	打包的执行步骤配置，由一系列Action组成。
data-dir	工程内使用数据。比如生成代码的模板，单词库，代码片段。
action-config	actions配置，打包开始会自动加载配置中的Action类。默认会加载打包工具的actions下的配置actions.json，并载入相应的Action。

目录结构

/ actions　存放执行动作的目录，组成打包步骤。 clang　python版本的clang接口，用于处理c++,objc文件。 cparser　封装了处理c++和objc文件，把ast转成类和函数。
data　保存工具中使用的数据。如模板，代码段，单词库。 doc　文档说明。 garbage_code　垃圾代码功能模块。 libclang　clang的动态库。 resource　资源处理模块。 rules　封装关系运算。

已经实现动作

文件操作

- 复制文件：copy_files(from,to) from,to可以为文件和文件夹。
- 删除文件：delete_files(files) files一个数组包含文件和文件夹
- 修改文件：modify_files(files) files是一个修改信息的数组。 结构为：{ “operation”:修改类型。 “keys”:关键字用于insert。操作内容为用keys找到的位置。 “olds”,”news”:用于replace。操作内容为olds指定的内容。 “froms”,”tos”:用于search_replace和remove。操作内容为froms到tos之间的所有内容。 “words”|”words_file”:用于search_replace替换的内容。 } 操作类型有：insert,insert_before,replace,search_replace,search_replace_to_end,remove。

xcode工程

- 修改工程：xcode_reanme(xcode_project_path,package_id,target_name,display_name,xcode_project_name,product_name) 主要修改xcode工程名，包名，显示名，生成的二进制名。
- 设置签名:xcode_set_sign(xcode_project_path,code_sign_identity,provisioning_profile,development_team,provisioning_profile_uuid,code_sign_entitlements) 设置xcode的开发者账相关信息，可以在一个xcode工程里把开发者账号设置好，打开project.xcodeproj里获取相关数据。
- 生成ipa:xcode_build_app(xcode_project_path,target,configuration,sdk,out_put) 一键生成ipa包。configuration通常为release,sdk为iphoneos.
- 生成archive:xcode_build_archive(xcode_project_path,scheme,configuration,out_put) 一键生成archive包。configuration通常为release,sdk为iphoneos.

混淆资源

- 路径转成hash值：obfuscate_resources. 把文件转成路径的32位hash值，在生成hash值的时候使用混淆因子使得每次生成的文件路径的hash值不同。这里的路径是相对路径，相对处理的根目录。为了避免根目录下文件太多，使用hash值的第一个字符作为目录。 参数：
    - res_path:要混淆的目录。
    - sub_dirs:子目录。在计算路径的时候，把子目录设置的的当作根目录，计算其下面文件的相对路径。
    - ignore_dirs:忽略目录。直接跳过这些目录不处理。
    - remove_source:是否删除原资源。只有在输出目录和要处理的目录不同时才有意义。
    - out_path：输出目录。保存混淆后的文件的地方。
- 目录映射：mapping_resources. 随机生成一批目录(多级)，把原目录下的文件，重命名，随机分布到新生成的目录内。这样比用存hash表示的文件系统更有意义。 参数：
    - res_path: 要混淆的目录。
    - out_path:输出目录。
    - mapping_file:保存映射数据。
    - min_level,max_level：定义生成的目录的层级。
    - min_dir_counts,max_dir_counts:定义每个层级最大目录数。这里是层级的数组，如果层级大于数组长度，则取最后一个数据。
    - remove_source:是否删除原文件。

    - with_ext:是否保留扩展名。
    - ignore_root:是否生成一个根目录。
    - save_json:是否把映射数据保存成json。
    - save_plist:是否把映射数据保存成plist。
    - include_rules,exclude_rules:文件的处理规则。
    - clean：是否清除目标目录。

- 合并映射数据:merge_mapping_file(files,out_path) 如果有多个目录要分开映射，会生成多个映射数据。如果需要统一处理，则合并成一个。

- 文件加密:crypt_files(from,to,key,sign,include,exclude) 使用xxtea对文件进行加密。可以定义包含和排除特定文件。

- 生成垃圾文件:generate_files 在指定位置随机生成文件，用于迷惑苹果扫苗文件系统。

混淆代码

- 生成c++类:generate_cpp_code 生成c++的头文件和源文件，头文件定义类，源文件实现类的方法。每个方法随机调用类的属性和其它方法。 参数：
    - out_dir:输出目录。
    - tpl_dir:代码的模板文件。
    - xcode_project_path:生成代码后。把生成的文件加入xcode项目中。并设置包含目录。
    - group_name:xcode工程中的group名称。如果不写，会随机生成。
    - exec_code_file_path:生成一个函数，把每个生成的类都调用一遍。
    - min_generate_file_count,max_generate_file_count:生成的文件数。
    - min_generate_field_count,max_generate_field_count:生成类的属性数量。
    - min_generate_method_count,max_generate_method_count:生成类的方法数量。
    - min_parameter_count,max_ parameter_count:每个方法的参数数量。
    - call_others:是否调用其它函数。
- 生成objc代码：generate_objc_code. 和生成c++一样的功能，只是类名和方法名是用单词组成，而不是简单的随机。参数也和c++一样。
- 插入c++代码：inject_cpp_code(files,tpl_dir,clang_args) 用clang分析所有c++文件，得到有实现的函数，随机在函数内部插入一段代码，并且代码不会被编译器优化掉。