**YAMLWeave - C语言自动插桩软件**

**用户操作手册**

1. **功能界面介绍顺序：登陆页、首页、主页、各个功能界面及其子界面+详细文字介绍；**
2. **操作界面截图规则：整个操作界面完整截图（包含外边框）、截图清晰、有数据参数（测试数据是可以的）、**
3. **程序界面所展示的产品名称和本次填写《信息采集表》中的软件“软件全称”或者“软件简称”一致；程序登记主体等版权声明需要和本次登记著作权人一致。**

**目 录**

[1. 引言 5](#_Toc133668697)

[1.1 目的 5](#_Toc133668698)

[1.2 背景、应用目的与功能概述 5](#_Toc133668699)

[1.3 术语和缩略语 5](#_Toc133668700)

[1.4 参考资料 5](#_Toc133668701)

[2. 系统运行环境 5](#_Toc133668702)

[2.1系统硬件环境 5](#_Toc133668703)

[2.2系统软件环境 5](#_Toc133668704)

[3. 系统操作说明 5](#_Toc133668705)

[3.1 系统操作流程 5](#_Toc133668706)

[3.2 系统的启动 5](#_Toc133668707)

[3.3 主要功能操作说明 6](#_Toc133668708)

# 引言

## 目的

本用户操作手册的编写目的主要是为YAMLWeave软件著作权登记提供完整的功能实现证明和技术成果固化，通过详细的界面截图和操作流程展示软件的独创性、完整性和商业化成熟度，同时确保产品名称和版权声明与登记信息保持一致。预期读者知识产权代理人/律师（审查文档合规性）、技术专家/评审员（评估技术先进性和创新点），该手册将作为软件著作权保护的核心证据材料，为后续的知识产权维护、技术转化和商业应用提供法律和技术支撑。

## 背景、应用目的与功能概述

系统名称

YAMLWeave - C代码插桩工具

系统任务提出背景

- 传统C语言测试插桩痛点：手动插入测试桩代码效率低，源代码污染严重

- 自动化测试需求：大型项目需要高效管理大量测试用例和桩代码

- 代码配置分离趋势：现代开发要求桩代码集中管理和复用

开发者

© 2025 刘浩洋

系统完成的主要功能

1. 双模式插桩：传统模式（注释内嵌代码）+ 分离模式（YAML配置管理）

2. 多文件批量处理：自动扫描.c/.h文件，支持跨文件测试用例组织

3. 智能备份管理：自动备份原项目，插桩结果保存到独立目录

4. YAML配置管理：结构化桩代码定义，支持复用和集中维护

5. 可视化界面：Tkinter图形界面，实时进度显示和日志输出

6. 详细统计日志：完整处理记录，支持日志导出和历史查询

7. 反向YAML生成：从传统注释生成YAML配置，便于模式迁移

8. 智能错误处理：缺失配置提示，异常恢复和详细错误信息

## 术语和缩略语

YAML - YAML Ain't Markup Language（YAML不是标记语言）

GUI - Graphical User Interface（图形用户界面）

UI - User Interface（用户界面）

CLI - Command Line Interface（命令行接口）

API - Application Programming Interface（应用程序编程接口）

TC - Test Case（测试用例）

ID - Identifier（标识符）

OS - Operating System（操作系统）

IO - Input/Output（输入输出）

UTF - Unicode Transformation Format（Unicode转换格式）

GBK - Guo Biao Kuozhan（国标扩展，中文编码标准）

ASCII - American Standard Code for Information Interchange（美国信息交换标准代码）

JSON - JavaScript Object Notation（JavaScript对象表示法）

XML - eXtensible Markup Language（可扩展标记语言）

HTTP - HyperText Transfer Protocol（超文本传输协议）

URL - Uniform Resource Locator（统一资源定位符）

TCP - Transmission Control Protocol（传输控制协议）

DLL - Dynamic Link Library（动态链接库）

【中文专门术语定义】

插桩 - 在程序代码中插入额外的代码片段，用于测试、调试或监控程序执行状态的技术手段

桩代码 - 为了测试目的而插入到程序中的代码片段，通常用于模拟特定的执行条件或验证程序行为

锚点 - 在源代码中标记的特定位置标识符，用于指示桩代码的插入位置

分离模式 - 将桩代码与源代码完全分离管理的工作模式，桩代码存储在外部配置文件中

传统模式 - 将桩代码直接嵌入在源代码注释中的工作模式

三级索引 - 测试用例ID-步骤ID-代码段ID的层次化组织结构

代码污染 - 在源代码中混入非业务逻辑代码，影响代码可读性和维护性的现象

自动插桩 - 通过工具自动化完成代码插桩过程，无需手动编辑源文件

跨文件测试 - 测试用例跨越多个源文件进行组织和执行的测试方式

编码检测 - 自动识别文本文件字符编码格式的技术

时间戳备份 - 使用时间标识创建的文件备份机制

模块化设计 - 将软件系统分解为独立、可重用模块的设计方法

解耦 - 降低软件模块间依赖关系，提高系统灵活性的设计原则

回调函数 - 被传递给其他函数作为参数的函数，在特定条件下被调用

正则表达式 - 用于匹配字符串模式的形式化语言

文件I/O - 文件输入输出操作

多线程 - 程序同时执行多个执行路径的并发处理技术

异常处理 - 程序运行时错误的捕获和处理机制

版本控制 - 管理文件变更历史的系统

软件打包 - 将软件及其依赖项打包成可执行文件的过程

【技术框架和工具】

Tkinter - Python内置的图形用户界面开发工具包

PyInstaller - Python程序打包为独立可执行文件的工具

PyYAML - Python解析和生成YAML文件的库

chardet - Python字符编码检测库

logging - Python内置日志记录模块

threading - Python多线程处理模块

subprocess - Python子进程管理模块

pathlib - Python路径操作库

datetime - Python日期时间处理模块

shutil - Python高级文件操作模块

glob - Python文件路径模式匹配模块

importlib - Python动态导入模块

## 参考资料

C语言标准

* 标题：ISO/IEC 9899:2018 - Programming languages — C
* 文件编号：ISO/IEC 9899:2018
* 发表日期：2018年
* 出版单位：ISO/IEC
* 来源：https://www.iso.org/standard/74528.html

UTF-8编码标准

* 标题：RFC 3629 - UTF-8, a transformation format of ISO 10646
* 文件编号：RFC 3629
* 发表日期：2003年11月
* 出版单位：IETF
* 来源：<https://tools.ietf.org/html/rfc3629>

# 系统运行环境

## 2.1系统硬件环境

CPU：Intel Core i3 2.0GHz及以上（文本处理负载较轻）

内存：2GB以上（软件运行时内存占用约50-100MB）

硬盘：50MB可用空间（可执行文件约30MB + 日志文件）

## 2.2系统软件环境

软件运行支撑环境/支持软件：

Python 3.7+运行时（源码模式）或独立可执行程序（打包模式）

# 系统操作说明

## 系统操作流程

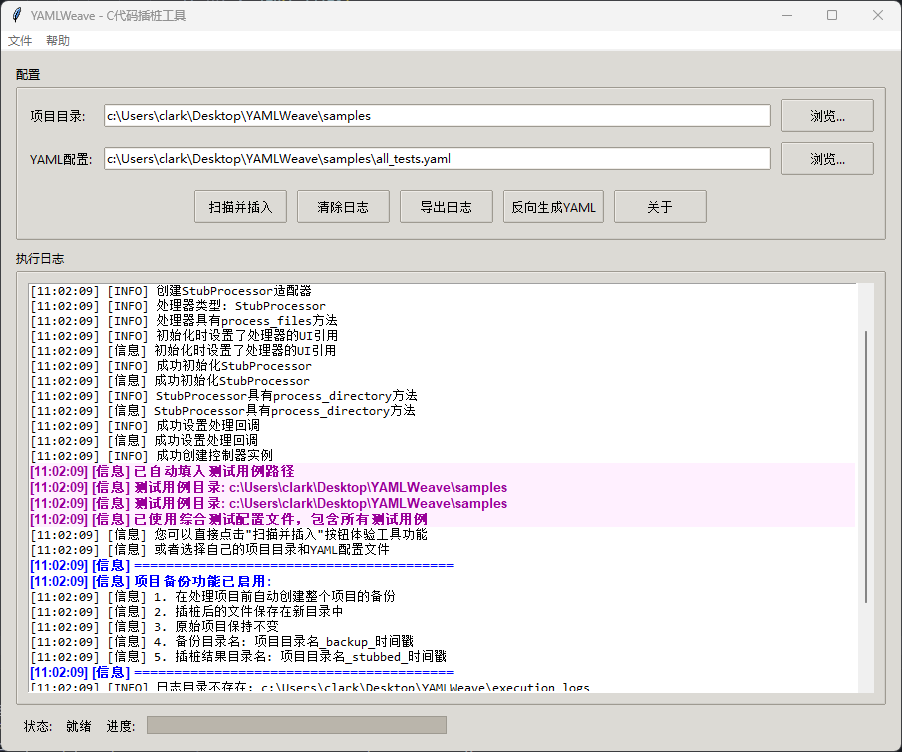
YAMLWeave系统采用图形化界面操作方式，支持传统模式和分离模式两种插桩方式。以下为详细的系统操作流程：

3.1.1 程序启动

- 双击运行YAMLWeave.exe程序文件

- 系统自动初始化日志记录器，创建日志目录

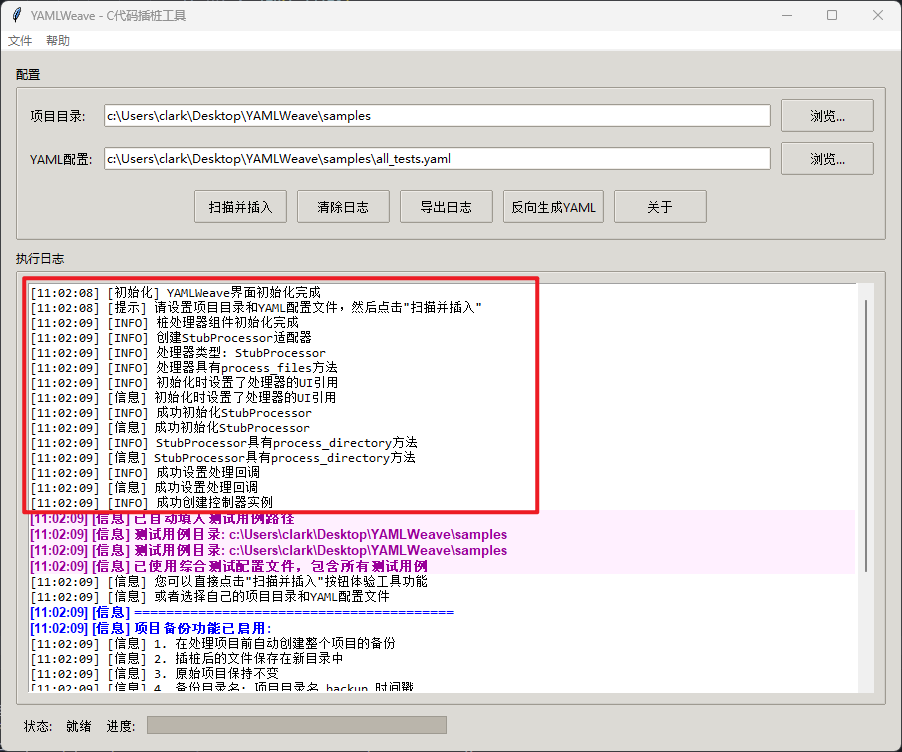
- 加载图形化用户界面，显示主操作窗口



2. 环境检查

- 系统自动检查运行环境和依赖项

- 初始化核心处理模块（StubProcessor、YamlStubHandler、CommentHandler）

- 在日志窗口显示"[初始化] YAMLWeave界面初始化完成"  


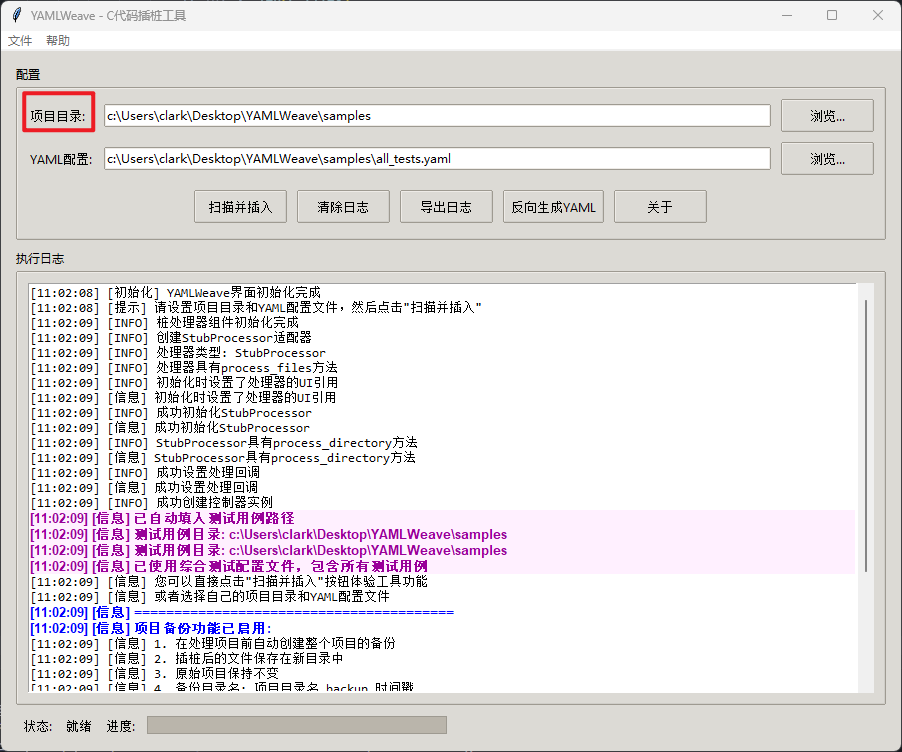
3.1.2 项目配置流程

1. 选择项目目录

- 点击"项目目录"行的"浏览..."按钮

- 在弹出的文件夹选择对话框中选择包含C源文件的项目目录

- 系统验证目录有效性，日志窗口显示"[信息] 已设置项目目录: {目录路径}"

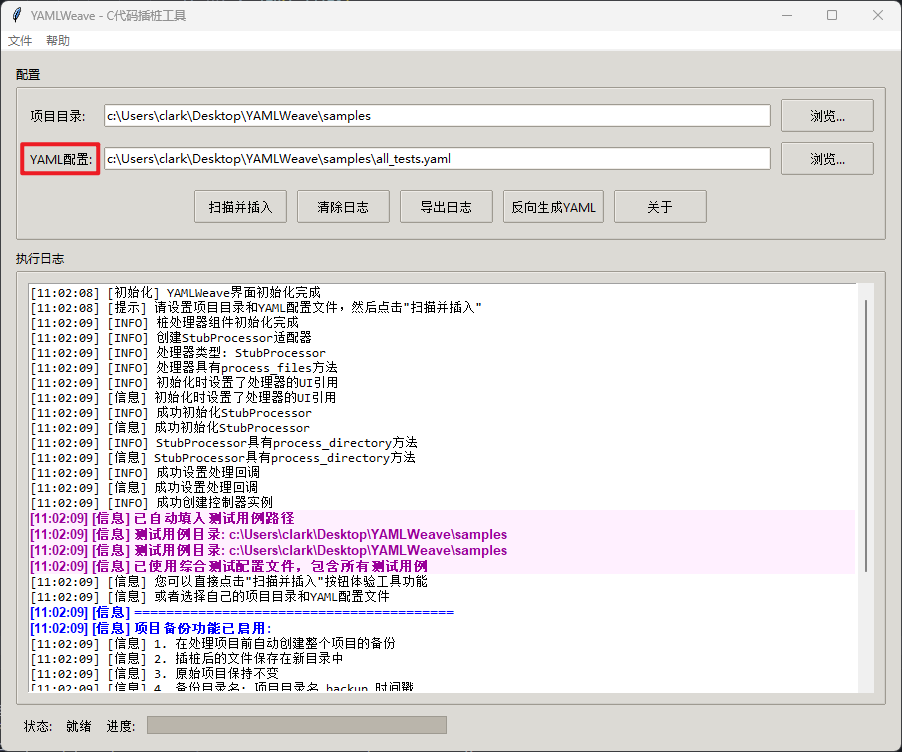


2. 选择YAML配置文件（分离模式必选，传统模式可选）

- 点击"YAML配置"行的"浏览..."按钮

- 选择.yaml或.yml格式的配置文件

- 系统加载并验证YAML文件格式，日志窗口显示"[信息] 已设置YAML配置:



3.1.3 插桩执行流程

1. 扫描并插入操作

- 点击"扫描并插入"按钮启动插桩处理

- 系统执行以下步骤：

Step 1: 文件发现阶段

- 递归扫描项目目录，查找所有.c和.h文件

- 在日志中显示发现的文件列表

- 更新进度条显示扫描进度

Step 2: 备份创建阶段

- 自动创建备份目录：{原目录名}\_backup\_{时间戳}

- 完整复制项目目录结构和所有文件到备份目录

- 日志显示"[备份] 备份目录创建完成：{备份路径}"

Step 3: 文件解析阶段

- 逐个解析C文件，查找插桩锚点或注释标记

- 根据工作模式解析桩代码：

传统模式：从注释中的code:字段提取桩代码

分离模式：根据锚点标识从YAML配置中查找对应桩代码

Step 4: 代码插入阶段

- 在识别的插入点位置插入桩代码

- 为插入的每行代码添加标识注释：// 通过桩插入

- 实时更新日志显示插入详情：

[找到] 文件: module1/validation.c，锚点: TC001 STEP1 validate\_input (第15行)

[插入] 桩代码: TC001.STEP1.validate\_input → 3行代码

Step 5: 结果生成阶段

- 创建结果目录：{原目录名}\_stubbed\_{时间戳}

- 将处理后的文件保存到结果目录

- 保持原有目录结构，仅修改包含插桩的文件

3.1.4 执行结果反馈

1. 实时日志显示

- 在界面下方的日志窗口实时显示处理进度

- 使用不同颜色标识不同类型的信息：

蓝色：文件信息

绿色：成功插入

紫色：发现锚点

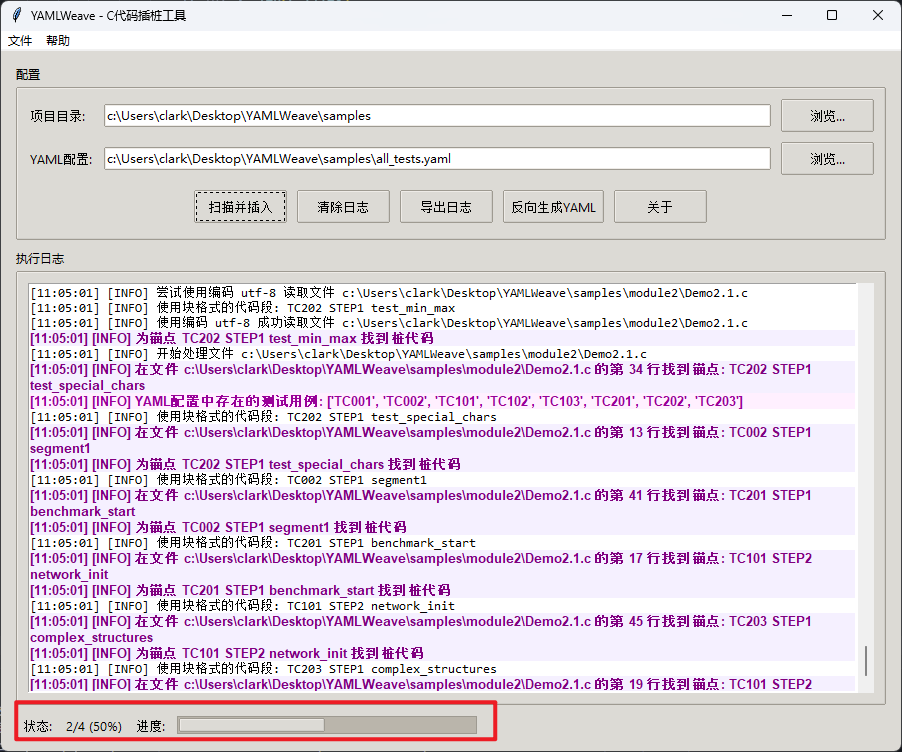
红色：错误信息

橙色：警告信息

2. 进度条更新

- 底部进度条显示当前处理进度

- 状态栏显示当前操作状态



3. 统计信息总结

- 处理完成后显示详细统计：

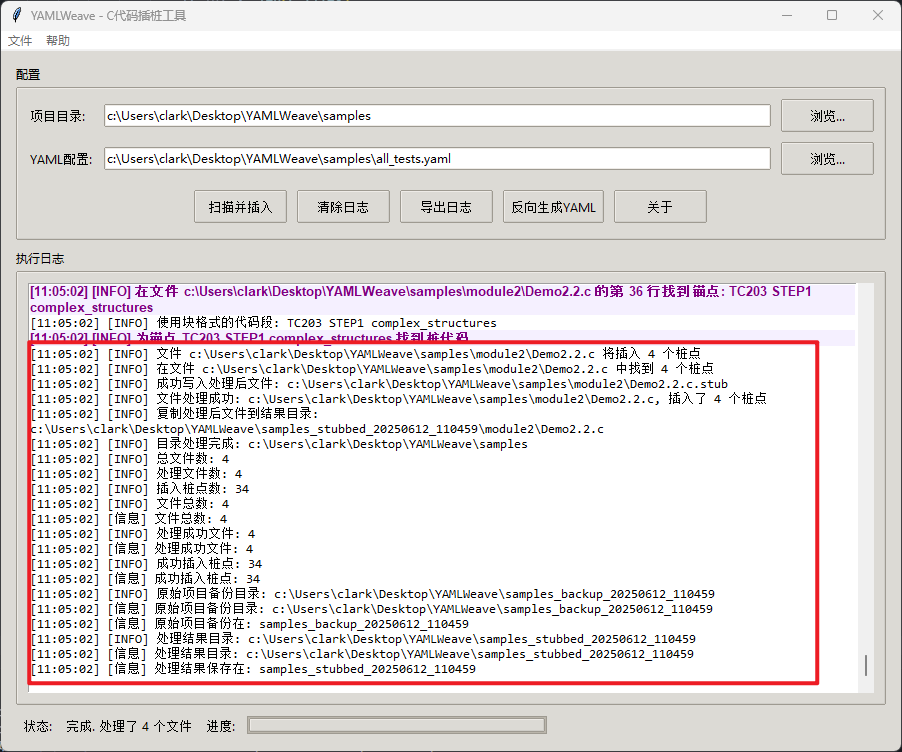
[统计] 处理完成！

[统计] 总文件数: 8

[统计] 处理文件数: 5

[统计] 成功插入桩点数: 12

[统计] 缺失桩点数: 2



3.1.5 辅助功能操作

1. 清除日志

- 点击"清除日志"按钮清空日志窗口内容

- 便于重新开始新的插桩操作

2. 导出日志

- 点击"导出日志"按钮将当前日志内容保存到文本文件

- 便于问题排查和结果分析

3. 反向生成YAML

- 点击"反向生成YAML"按钮

- 从现有的传统模式注释中提取桩代码，生成YAML配置文件

- 支持从传统模式转换到分离模式

3.1.6 错误处理流程

1. 输入验证

- 未选择项目目录时弹出警告："请选择项目目录"

- 目录不存在时显示错误信息

- 分离模式下未选择YAML文件时给出提示

2. 处理异常

- 文件读取失败时在日志中显示具体错误信息

- YAML格式错误时提供详细的错误位置

- 权限不足时提示用户以管理员身份运行

3. 结果验证

- 处理完成后验证生成文件的完整性

- 检查备份目录是否创建成功

- 统计并报告缺失的桩代码数量

3.1.7 退出流程

1. 正常退出

- 点击窗口右上角关闭按钮或菜单栏"文件→退出"

- 系统自动保存日志文件到logs\_{时间戳}/yamlweave.log

2. 异常处理

- 处理过程中可通过关闭窗口中断操作

- 已创建的备份文件会保留，方便恢复

3.1.8 针对反向生成YAML功能的操作说明

1. 准备阶段

确保项目目录中包含已经插桩的C文件（带有// 通过桩插入标记的文件）

启动YAMLWeave GUI界面

2. 选择项目目录

在GUI界面中点击"浏览..."按钮

选择包含已插桩C文件的项目根目录

确保目录路径显示在"项目目录"输入框中

3. 启动反向生成

点击界面上的"反向生成YAML"按钮

系统会弹出文件保存对话框

4. 选择输出位置

在文件保存对话框中：

选择YAML文件的保存位置

输入文件名（默认为reversed.yaml）

确认文件类型为.yaml或.yml

5. 执行提取过程

点击"保存"开始提取过程

系统会自动：

扫描项目目录中的所有.c和.h文件

识别带有锚点标记的代码行

提取桩代码内容

按照标准YAML格式组织数据

6. 查看执行结果

在日志区域查看提取进度和结果

成功完成后会显示[信息] 成功导出YAML: [文件路径]

如有错误会显示具体的错误信息

3.1.8 操作流程总览

完整的系统操作流程如下：

1. 启动YAMLWeave程序

2. 界面初始化完成

3. 选择项目目录

4. 选择YAML配置文件（分离模式必选，传统模式可选）

- 如选择分离模式：加载YAML配置

- 如选择传统模式：直接进入处理

5. 点击"扫描并插入"按钮

6. 文件发现扫描阶段

7. 创建备份目录

8. 逐个处理C文件

9. 解析插桩标记

10. 根据工作模式判断：

- 传统模式：从注释提取桩代码

- 分离模式：从YAML查找桩代码

11. 插入桩代码到指定位置

12. 生成结果文件

13. 重复步骤8-12直到所有文件处理完成

14. 显示统计结果

15. 操作完成

3.1.9 最佳实践建议

1. 使用前准备

- 建议先对重要的源代码进行手动备份

- 确保项目目录具有读写权限

- 为复杂项目准备详细的YAML配置文件

2. 分离模式推荐

- 对于大型项目，建议使用分离模式以保持代码整洁

- YAML配置文件应与源代码同步维护

- 建议为不同测试场景创建独立的YAML配置文件

3. 结果验证

- 处理完成后检查备份目录和结果目录是否正确生成

- 通过编译测试验证插桩后的代码语法正确性

- 建议使用版本控制系统管理插桩前后的代码变更

通过以上详细的操作流程，用户可以高效地使用YAMLWeave工具完成C代码的自动插桩工作，既支持快速的传统模式插桩，也支持更加规范的分离模式管理。