**流程图总览**

textCopy Code

[源代码符号]

↓ 预处理 → 宏/头文件符号替换（文本级操作）

[预处理后符号]

↓ 编译 → 符号表生成（记录类型/作用域/地址）

[带元信息的符号]

↓ 汇编 → 符号转为机器码偏移地址

[可重定位符号]

↓ 链接 → 符号合并/地址重定位/外部解析

[最终可执行符号]

**分阶段解析**

**1. 预处理阶段：符号的文本级操作**

▌‌**符号变化特点**‌：

* 宏符号直接替换（如#define PI 3.14 → 所有PI替换为3.14）
* 头文件符号展开（如#include <stdio.h> → 插入printf的声明）
* ‌**不生成符号表**‌，仅生成纯文本代码

▌示例：

cCopy Code

*// 预处理前*

#define MAX 100

int arr[MAX]; *// → 预处理后替换为 int arr[100];*

**2. 编译阶段：符号表生成与语义绑定**

▌‌**符号变化特点**‌：

* 生成符号表（记录符号名称、类型、作用域、临时地址等）
* 符号分类：

markdownCopy Code

- 全局符号（Global）：`int global\_var;`

- 局部符号（Local）：函数内部变量

- 外部符号（External）：`extern void func();`

* 语义错误检测（如重复定义、类型不匹配）

▌符号表示例（伪代码）：

plaintextCopy Code

符号名：global\_var

类型：int

作用域：全局

存储位置：数据段偏移+0x00

状态：已定义

**3. 汇编阶段：符号到机器地址的映射**

▌‌**符号变化特点**‌：

* 符号表中的逻辑地址转为‌**可重定位地址**‌（如mov eax, [global\_var] → 机器码中使用偏移量）
* 符号可见性标记：

markdownCopy Code

- `.global func` → 允许其他模块访问

- `.local var` → 仅当前文件可见

* 生成目标文件符号表（ELF格式的.symtab节）

▌使用objdump查看符号地址：

bashCopy Code

objdump -t main.o

*# 输出示例：*

00000000 g O .data 00000004 global\_var *# 偏移地址0x00*

00000000 g F .text 0000001a main

**4. 链接阶段：符号解析与全局整合**

▌‌**符号变化特点**‌：

1. ‌**符号合并**‌：
   * 消除重复全局符号（同名全局变量/函数仅保留一个）
   * 示例错误：multiple definition of 'global\_var'
2. ‌**地址重定位**‌：
   * 分配绝对内存地址（如global\_var从偏移0x00 → 绝对地址0x08049000）
3. ‌**外部符号解析**‌：
   * 链接库文件填补未定义符号（如printf从U标记转为T标记）

▌动态链接特殊处理：

* 符号延迟绑定（PLT/GOT表技术）
* 示例：readelf -d app 查看动态符号表

**符号状态演化表**

| **阶段** | **符号类型** | **符号地址** | **关键操作** |
| --- | --- | --- | --- |
| 预处理 | 文本标识符 | 无 | 宏替换/头文件展开 |
| 编译 | 带类型/作用域符号 | 逻辑地址 | 符号表生成/语义检查 |
| 汇编 | 可重定位符号 | 段内偏移量 | 生成机器码/可见性标记 |
| 链接 | 绝对地址符号 | 虚拟内存地址 | 合并/重定位/解析外部 |

**实验验证（Linux环境）**

bashCopy Code

*# 1. 预处理后查看符号替换*

gcc -E main.c -o main.i

cat main.i | grep "替换后的符号"

*# 2. 编译阶段生成符号表（GCC调试选项）*

gcc -S main.c -fverbose-asm *# 生成带符号注释的汇编代码*

*# 3. 查看目标文件符号表*

readelf -s main.o *# 显示未链接符号状态*

*# 4. 对比链接前后符号地址*

nm main.o *# 链接前偏移地址*

nm app *# 链接后绝对地址*

**典型符号生命周期示例**

cCopy Code

*// file1.c*

extern void print(); *// 编译：标记为UND（未定义）*

int global = 10; *// 链接：分配地址0x08049000*

*// file2.c*

void print() { */\*...\*/* } *// 链接：地址重定位为0x08048000*

**总结**

* ‌**预处理**‌：符号仅文本替换，无结构化记录
* ‌**编译**‌：符号表核心生成阶段，绑定语义信息
* ‌**汇编**‌：符号转为机器码可处理的偏移地址
* ‌**链接**‌：符号全局唯一化并绑定物理内存地址