

第二讲：用好写好头文件

魏永明的 C 语言最佳实践课程

复习一下

- 头文件的作用：用于声明公用的函数、宏、数据类型或结构。
- 头文件里边通常包含：函数声明、内联函数的实现、宏、枚举常量、数据类型、结构体的定义，以及 API 接口描述文档。
- 头文件通常不包含：非内联函数的具体实现。
- 头文件分类：
 - 标准 C 库的头文件，通常含有 std 前缀，如 `stdio.h`、`stdlib.h` 等。
 - 操作系统或者 POSIX 标准相关的头文件，比如 `unistd.h`、`sys/uio.h` 等。
 - 平台特有头文件，如 `windows.h`。
 - 第三方库头文件，如 `zlib.h`。
 - 自定义头文件。

常用 C99 头文件

- `stdio.h`: 标准 IO 接口: `printf/scanf`
- `stdlib.h`: 通用工具: `atoi/malloc/free`
- `stdarg.h`: 可变参数处理: `va_list`
- `stdint.h`: 整数类型及各整数类型的极值
- `stddef.h`: 常用类型以及宏的定义, 如 `size_t`、`NULL` 等
- `stdbool.h`: `bool` 类型、`true`、`false` 等
- `string.h/strings.h`: 字符串操作函数, 如 `strcat/strcasecmp` 等
- `ctype.h`: 字符操作函数或宏, 如 `toupper`
- `time.h`: 时间和时区操作函数, 如 `time`
- `math.h`: 数学函数, 如 `M_PI/sin/cos`

常用 POSIX 标准头文件

- `unistd.h`: 类型定义及常见系统调动接口声明, 如 `pid_t`、`fork` 等
- `fcntl.h`: 文件控制接口, 如 `creat`
- `dirent.h`: 目录操作接口, 如 `opendir`
- `sys/*.h`: 系统类型等。

提问

C99 文件读写函数和
POSIX 文件读写函数
的区别



滥用头文件的坏处

- 拖慢编译速度。
- 破坏可移植性。
 - POSIX 标准头文件并不是所有平台上都可用。
 - 在 RTOS 或物联网小系统中，某些 C99 接口可能残缺，如标准 IO 相关接口。
- 容易造成命名污染。

用好、写好头文件的两大原则

- 最少包含
 - 只包含必需的头文件
 - 只包含会被多个源文件共用的内容
- 自立 (self-contained)
 - 任意一个头文件放到源文件的第一行，都不会出现未定义或者未声明的编译错误。

头文件误用范例

注意事项

- 防止重复包含
 - 重复包含保卫宏的命名规则
 - 为什么 C 头文件较少使用 `pragma once`?
- 外部函数和全局变量声明针对 C++ 做特别处理

示范项目说明

<https://github.com/VincentWei/best-practices-of-c>
<https://gitlab.fmsoft.cn/VincentWei/best-practices-of-c>

一般项目的四类头文件

- 对外头文件 (API)
- 全局内部头文件
- 模块内部头文件
- 构建系统生成的头文件

命名、路径和安装规范

- 头文件命名
- 目标安装路径
- 编译时的头文件搜索路径

用两个原则审视示范项目

提问

隐藏结构细节有什么
好处？



重要技巧

用编译时配置选项生成头文件

用编译时配置选项生成头文件

- 必要性
 - 方便用户
 - 方便维护
- 方法（以 cmake 为例）
 - 定义模板
 - 生成头文件
- 注意事项
 - 头文件搜索路径
 - 始终将项目头文件安装到自己的目录中
 - 安装最少的头文件到系统中

Q & A