# C语言技术管理规范文档

## 一、编码风格规范

### a. 强制规则

1. 缩进必须使用4个空格，禁止使用Tab字符。
2. 每行代码长度不得超过 100 个字符，必要时进行换行。
3. 每个源文件必须包含文件头注释，注明作者、日期、模块功能简介。
4. 所有标识符必须使用小写字母+下划线命名风格（snake\_case），不允许混用大小写（如 camelCase）。
5. 每个函数前必须添加注释，说明其功能、参数、返回值及副作用。
6. 必须使用头文件 include guard 或 #pragma once 防止重复包含。
7. 禁止使用魔法数字（magic numbers），所有常量必须使用宏或 const 常量定义。
8. 不允许使用 goto 除非处理严重错误时必须清理资源。
9. 不允许使用未初始化的变量。
10. 头文件中禁止定义函数实现（除内联函数外）。
11. 禁止在头文件中定义非static的全局变量。
12. 严禁在生产代码中使用 printf 或 fprintf(stderr, ...) 作为调试手段，需使用统一日志模块。
13. 禁止嵌套过深的 if/else（超过3层需重构）。
14. 所有函数必须显式声明返回类型，不允许省略 int 等隐式返回。
15. 所有函数参数、局部变量必须显式初始化。

### **b. 推荐规则**

1. 推荐在函数内部尽量采用 const 限定不可变变量。
2. 推荐每个函数长度不超过 50 行，若超过应拆分为子函数。
3. 推荐使用布尔表达式时避免使用隐式转换（例如 if (ptr) 应写成 if (ptr != NULL)）。
4. 推荐将所有头文件的引用按标准库、第三方库、自定义模块顺序排列，并在每组之间加空行。
5. 推荐每个模块提供 .h + .c 配对实现，防止头文件泄露私有实现。
6. 推荐在所有 switch 语句中添加 default 分支，即使暂时不做处理。
7. 推荐结构体字段对齐排列，成员注释写在字段右侧。
8. 推荐使用 typedef 定义结构体别名，如 typedef struct user\_t {...} user\_t;。
9. 推荐用 do { ... } while(0) 包裹多语句宏，避免宏展开错误。
10. 推荐每段逻辑用空行分隔，增加可读性。
11. 推荐所有指针在声明时用 type \*ptr 格式（星号靠近类型），便于统一识别。
12. 推荐注释使用英文为主，必要时使用双语注释。
13. 推荐关键语句（如资源释放、错误处理）使用注释强调其作用。
14. 推荐定义明确的模块边界与接口，不在.c文件之间相互调用私有函数。
15. 推荐将资源管理函数统一为 init() / destroy() 或 open() / close() 对。

### **c. 允许规则**

1. 允许使用宏进行平台相关适配，但必须有注释说明各平台差异。
2. 允许在特定场景使用全局变量，但必须注明其唯一性与线程安全策略。
3. 允许在项目内设置局部风格偏移（如测试代码允许使用 assert），但需写明理由。
4. 允许在 struct 内使用嵌套 struct 类型作为成员，但不宜超过2层嵌套。
5. 允许使用多重宏展开（宏套宏）构造简化语法糖，但需确保可读性和调试友好性。
6. 允许在函数声明中省略参数名（如 int foo(int, int);），仅限头文件中使用。
7. 允许在头文件中使用 static inline 函数提升性能，但不可过多或滥用。
8. 允许使用变量初始化列表（如 int arr[] = {1, 2, 3}），但推荐在复杂初始化中使用专门函数封装。
9. 允许在 debug 编译中引入 assert/assert.h，release 版本需确保无副作用。
10. 允许在测试代码中使用固定 seed 的伪随机函数进行一致性测试。

## **二、内存管理规范**

### **a. 强制规则**

1. 所有 malloc/calloc/realloc 分配的内存必须由 free 释放。
2. 禁止内存泄漏，所有分配必须成对释放，建议使用工具检测（如 valgrind）。
3. 禁止释放未初始化或已经释放的指针。
4. 禁止混用不同方式分配/释放的内存（如 malloc 对 free，new 对 delete）。
5. 禁止将 malloc 的返回值直接强转为其他指针类型（C 中无此必要）。

### **b. 推荐规则**

1. 推荐将内存释放写在 finally-style 的清理路径中，避免遗漏。
2. 推荐使用 memset 清空敏感数据（如密码）再释放内存。
3. 推荐封装资源管理模块，如统一写 buffer\_alloc() / buffer\_free() 管理内存。
4. 推荐使用结构体封装大块内存，统一释放逻辑。
5. 推荐对 malloc 结果做 NULL 检查，即使部分平台不会返回 NULL。

## **三、并发与线程安全规范**

**a. 强制规则**

1. 禁止使用非线程安全函数（如 strtok、gmtime），必须使用其线程安全版本（如 strtok\_r、gmtime\_r）。
2. 在多线程环境下禁止修改共享变量，除非加锁或使用原子操作。

### **b. 推荐规则**

1. 推荐使用 pthread 或统一封装的线程模块，避免直接调用平台API。
2. 推荐每个线程绑定明确任务，避免职责重叠。
3. 推荐使用 RAII 风格封装锁资源（如 mutex\_lock() / mutex\_unlock() 封装进 struct）。

## **四、测试与可维护性规范**

### **a. 强制规则**

1. 所有模块必须至少具备一组基本功能测试用例。
2. 禁止将测试代码与生产代码混杂，测试应独立于 tests/ 目录中。
3. 必须确保所有代码通过静态分析工具（如 cppcheck）检查无高危警告。

### **b. 推荐规则**

1. 推荐使用 CMocka、Unity 等测试框架编写单元测试。
2. 推荐每个模块代码覆盖率达到80%以上。
3. 推荐使用 Git pre-commit hook 进行代码格式检查与自动测试执行。

## 五、文档与交付规范

### **a. 强制规则**

1. 每个模块必须编写 README.md 说明接口、编译方法、依赖项等内容。
2. 所有 API 函数必须在 .h 文件中配套注释其用途、参数、返回值及异常情况。

### **b. 推荐规则**

1. 推荐在交付版本中添加版本号、构建时间、编译选项信息。
2. 推荐统一用 Doxygen 风格编写注释以生成文档。
3. 推荐文档与代码同步更新，禁止版本漂移。