第一讲：提高代码可读性

魏永明的 C 语言最佳实践课程

代码可读性的衡量标准

- 初级：能编译通过即可，这表明计算机能读懂你的代码。
- 中级：三个月之后，你自己还能看懂。
- 高级：接手你工作的其他码农能看得懂，且改得动。
- 神级：不需要修改和维护，可以运行“万年”的。

为什么有些代码看了就想吐?

坏代码的共同点

不好的编码风格

- 排版混乱
- 书写拥挤
- 混用不同的编码风格

坏代码的共同点

不好的命名

- 不正确的术语
- 不符合习惯
- 错误的时态
- 使用拼音
- 含有中文等特殊字符的文件名

坏代码的共同点

不好的注释

- 太多或太少
- 使用中文
- 注释太花哨

实例：随处可见的坏代码

编码风格到底规定什么？

排版规则

- 缩进、空格、换行、大括号位置等

编码风格到底规定什么？

命名规则

- K & R 规则：this_is_an_integer
- 匈牙利规则：iThisIsAnInteger
- 其他：比如宏、枚举常量、全局变量等的命名规则

编码风格到底规定什么？

其他规则

- 自定义类型的使用
- 条件编译的写法
- 注释的写法
- 经验写法

实例：Linux 内核的编码风格

有关C编码风格的常见争议

80 列红线

- 最佳实践：务必守好 80 列红线
  - 防止过多缩进嵌套，强制改进代码
  - 在大屏幕上，方便同时查看多个源文件文件。

有关C编码风格的常见争议

空格

- 最佳实践：Linux 内核编码风格
  - 单目运算符、++、--不加空格：
    int *p = (int *)&a;
    p++;
  - 函数名称，包括可按函数调用的关键词之后，不要加空格：
    call_me(sizeof(int), MAX(x, y));
  - 不要在行尾加空格。
  - 其他情形，如双目或多目运算符前后、关键词之后都要加空格：
    if (a && b) {
    }

有关C编码风格的常见争议

指针的星号位置

- 最佳实践：Linux 内核编码风格
  - void *get_context(struct node *node)

有关C编码风格的常见争议

下划线前缀

- 最佳实践：
  - 仅针对 extern 变量或者函数使用下划线前缀：

    extern size_t __total_mem_use;
  - 作用：主要用于防止命名污染

有关C编码风格的常见争议

何时使用 typedef

- 最佳实践：无
- 我的实践：
  - 仅在函数库的接口定义中使用 typedef
  - 仅对结构的指针使用 typedef，并使用 _t 后缀
  - 对枚举或者结构的类型定义名称使用驼峰命名法

有关C编码风格的常见争议

命名规则

- 最佳实践：倾向 K & R 规则，避免使用匈牙利命名法如 spName
- 我的实践：
  - 除接口外，内部使用 K & R 命名规则

有关C编码风格的常见争议

注释

- 最佳实践
  - 避免使用 C++ 注释： //
  - 巧用 XXX、FIXME、TODO 等短语
  - 使用 Doxygen 格式来撰写 API 文档说明

命名的艺术

一般性原则

- 使用正确的英文术语及其简写
  node, child, sibling, parent, root, first, last, next, previous, …
- 避免使用万金油名称：
  item、data

命名的艺术

一般性原则

- 正确使用时态：
  linked_list
- 正确使用单数、复数形式：
  nodes, children

命名的艺术

一般性原则

- 使用喜闻乐见的缩写：
  nr, sz, dbl, tri, len, max, min, buf, ver, id,
  prev, tmp, param, arg, argc, argv,
  conn, ctxt, err, …

命名的艺术

局部变量

- 简洁：
  循环：i, j, k；数量：n；长度：len；
  尺寸：sz；指针：p
  临时变量：tmp；临时缓冲区：buf；

命名的艺术

函数及全局变量的名称

- 接口：
  <type> <lib prefix>_<short phrase>(...)
- 文件内：
  static <type> <short phrase>(...)
- 模块间：
  <type> _<module_prefix>_<short phrase>(...)
  __attribute__((visibility("hidden")))

实例：hiBus 源代码及经典的 list_head 结构

实例：整理 bin2c.c 源文件

Q & A