内核编码风格

缩进：

- 8字符缩进
- 一行内不要有多条语句
- 不用空格来缩进
- 不要在行末有空格

长行长串分割：

- 行长小于80字符
- 大于80个字符的语句分成多行，子行总比父行短，大体上向右靠齐。
- 带有长参数列表的函数也是如此。

括号：

- 语句块把左括号放在行尾，右括号放在行首
- 函数的左右花括号都放在行首，第一列
- 右括号一般独立一行，除了后面跟着相同类型的语句块，比如else或者do while
- 单语句的时候不使用括号，if-else，有一个使用了，另一个也用

空格：

- 这些关键字后用空格：if, switch, case, for, do, while
- 这些关键字后不用空格：sizeof, typeof, alignof, __attribute__
- 不要在括住的表达式两边使用空格，即左括号右端和右括号左端不用
- 函数返回值是指针类型是，'*'号紧跟函数名，定义指针类型变量也是
- 在下列二元三元运算符周围使用空格
  - = + - < > * / % | & ^ <= >= == != ? :
- 在下列一元运算符周围不使用空格
  - & * + - ~ ! sizeof typeof alignof __attribute__ defined
- 不在'++', '--'前后使用空格，不再'.'和'->'周围使用空格

命名规则：

- 全局变量必须是描述性的名字
- 全局函数也是
- 本地变量简短，切题。

Typedefs：

- 不要对结构体和指针使用typedef
- 使用typedef的场合
  - 完全不透明的对象（使用typedef来隐藏实际对象）
  - 清楚的整数数据类型，例如u8/u16/u32等
  - 当创建一个新类型使用语法检查的时候（不太懂）
  - 和标准C99相同的新类型
  - 可在用户空间安全使用的类型（还是不太懂）

函数：

- 函数应该简短而漂亮，函数应该为一或者两屏文本（80×24屏）
- 一个函数只完成一件事情，并且漂亮的完成。
- 函数最大长度和函数的复杂度以及缩进级数成反比。理论上很简单的函数你可以写的很长很长，比如多种情况的case语句
- 复杂的函数尽量遵守限制，并且使用帮助函数，取个好名字
- 函数的本地变量不应超过5-10个
- 使用空行分隔不同的函数
- 如果函数需要导出，EXPORT宏紧跟在函数结束的大括号后
- 函数原型中，包含函数名和类型

集中的函数退出途径

- goto语句，当一个函数从多个位置退出并且做一些通用的清洁工作

注释：

- 永远不要在注释中解释你的程序怎么运行，写好的代码比解释差的代码好
- 注释说明你的代码是什么而不是怎么做
- 尽量不要再一个函数体内部注释，在函数之前说明。
- 多行注释的风格如下

```
/*
 * This is the preferred style for multi-line
 * comments in the Linux kernel source code.
 * Please use it consistently.
 *
 * Description:  A column of asterisks on the left side,
 * with beginning and ending almost-blank lines.
 */
```

- 注释数据也是很重要的，尽量一行声明一个数据，以留出空间可以注释

格式化代码：

Kconfig 文件：

数据结构：

- 如果一个数据结构在创建和销毁它的单线程执行环境外可见，那么它必须有一个引用计数器，内核没有垃圾收集器，你需要它记录对这个数据结构的使用
- 引用计数可以避免加锁，允许多个用户平行访问该数据结构。
- 上锁不能取代引用计数，上锁是为了保持数据结构的一致性，引用计数是一种内存管理技巧，通常二者都需要
- 很多数据结构实际上有2级引用计数，它们通常有不同类的用户，子类计数器统计子类用户的数量，每当子类计数器减至零时，全局计数器减一

Macro，Enum and RTL

- 定义常量和ENUM中的标签需要大写

- 当定义几个相关常量时，尽量使用Enum
- 宏名字要大写，但形如函数的宏名可小写
- 如果能写成内联函数，就不要写成像函数的宏
- 含有多个语句的宏应该被包含在一个do-while代码块里
- 使用宏的注意事项
  - 避免使用影响控制流程的宏
  - 避免使用依赖于一个固定名字的本地变量的宏
  - 避免使用作为左值的带参数的宏
  - 牢记优先级

打印内核消息：

- 不要拼写错误
- 不必句号结束
- 在小括号里打印数字(%d)没有任何价值
- 当DEBUG符号没有被定义的时候，调试信息不应该被编译进内核

内存分配：

- 内核提供的一般用途的内存分配函数
  - kmalloc()
  - kzalloc()
  - kcalloc()
  - vmalloc()
- 最好使用这种形式传递数据结构大小
  ```
  p = kmalloc(sizeof(*p), ...);
  ```
- 强制转换一个void指针返回值是多余的，C会处理这件事情

内联的弊病：

- 基本原则是，如果一个函数有3行以上，那就不要内联。
- 例外，如果某个参数是编译时常量，而且编译时能优化掉大部分代码，那可以内联

函数返回值和命名：

- 如果返回表示函数执行成功或者失败，可以标识为一个错误代码整数或者成功布尔值
- 如果函数的名字是一个动作或者强制性命令，那么这个函数应该返回错误代码整数
- 如果函数的名字是一个判断，那么函数应该返回一个成功布尔值
- 所有导出的函数都必须遵守这个规则，所有的公共函数亦是，私有函数推荐如此
- 如果是返回实际计算结果，则以返回正常范围之外的结果表示出错，比如NULL。

不要重新发明内核宏

- include/linux/kernel.h中包含了大量的宏，使用它们