代码书写规范

1 缩进

1) 缩进采用 8 个字符
2) 在 switch 语句中消除多级缩进的首选的方式是让“switch”和从属于它的“case”标签对齐于同一列，而不要“两次缩进”“case”标签。
3) 不要把多个语句放在一行里，除非你有什么东西要隐藏：

```
if (condition) do_this;
do_something_everytime;
```

4) 不要在一行里放多个赋值语句。
5) 除了注释、文档和 Kconfig 之外，不要使用空格来缩进。

2 把长的行和字符串打散

1) 每一行的长度的限制是 80 列
2) 长于 80 列的语句要打散成有意义的片段。

3 大括号的放置

1) 结束大括号独自占据一行，除非它后面跟着同一个语句的剩余部分，也就是 do 语句中的“while”或者 if 语句中的“else”，像这样：

```
do {
body of do-loop
} while (condition);
```

和

```
if (x == y) {
..
} else if (x > y) {
...
} else {
....
}
```

2) 函数的起始大括号放置于下一行的开头，所以：

```
int function(int x)
{
body of function
}
```

4 空格的放置

1) 不要在小括号里的表达式两侧加空格：

   s = sizeof(struct file);

2) 当声明指针类型或者返回指针类型的函数时，“*”的首选使用方式是使之靠近变量名或者函数名，而不是靠近类型名。例子：

   char *linux_banner;
   unsigned long long memparse(char *ptr, char **retptr);
   char *match_strdup(substring_t *s);

3) 在大多数二元和三元操作符两侧使用一个空格，例如下面所有这些操作符：

   = + - < > * / % | & ^ <= >= == != ? :

   但是一元操作符后不要加空格：

   & * + - ~ ! sizeof typeof alignof __attribute__ defined

   后缀自加和自减一元操作符前不加空格：

   ++ --

   前缀自加和自减一元操作符后不加空格：

   ++ --

   “.”和“->”结构体成员操作符前后不加空格。

4) 不要在行尾留空白。

5) 这些关键字之后放一个空格：

   if, switch, case, for, do, while

   但是不要在 sizeof、typeof、alignof 或者__attribute__这些关键字之后放空格。例如，

   s = sizeof(struct file);

5 Typedef

不要使用类似“vps_t”之类的东西。

对结构体和指针使用 typedef 是一个错误。当你在代码里看到：

vps_t a;

这代表什么意思呢？

相反，如果是这样

struct virtual_container *a;

你就知道“a”是什么了。

6 注释

长（多行）的首选注释风格是：

```
/*
* 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.
*/
```

7 宏，枚举和 RTL

1) 用于定义常量的宏的名字及枚举里的标签需要大写。

```
#define CONSTANT 0x12345
```

在定义几个相关的常量时，最好用枚举。一般的，如果能写成内联函数就不要写成像函数的宏。

2) 含有多个语句的宏应该被包含在一个 do-while 代码块里：

```
#define macrofun(a, b, c)            \
do {                    \
if (a == 5)             \
do_this(b, c);          \
} while (0)
```

3) 不要定义依赖于一个固定名字的本地变量的宏：

```
#define FOO(val) bar(index, val)
```

4) 不要定义作为左值的带参数的宏： FOO(x) = y；如果有人把 FOO 变成一个内联函数的话，这种用法就会出错了。

5) 不要忘记了优先级：使用表达式定义常量的宏必须将表达式置于一对小括号之内。带参数的宏也要注意此类问题。

```
#define CONSTANT 0x4000
#define CONSTEXP (CONSTANT | 3)
```