Code Programming Specifications

版本历史记录

修改日期	版本	修改人	修改内容
2024/01/03	初版	Xiaoyibar	创建文档

目录

前言

在阅读一些不注重规范的代码时会看的人头大，很不容易摸清楚程序逻辑。所以在阅读拥有良好编编程习惯的人编写的代码时候 很容易看明白。代码就是给人看的，代码如果整洁、清晰、命名规范别人看起来就很容易。

本文按我自己喜欢的风格去做一个编程习惯的规范，约束自己。

1. 文件目录结构

1.1 头文件

- 【规则】防止头文件呗重复引用 必须 #ifndef/#define/#emdif
- 【规则】引用标准 C 库 <> 尖括号 引用自己或其他 “ ”双引号
- 【规则】头文件只声明不定义
- 【规则】头文件中尽量不要使用全局变量

```
/* 头文件注释说明 */
#ifndef __TEXT_H
#define __TEXT_H

#include <标准库>
#include “非标准库”

#define 宏定义

/* 枚举/结构体等类型声明 */
enum
{
    NONE_TEST = 0,
    SHORT_FRAME_TEST,
    LONG_FRAME_TEST
};

/* 函数声明 */
extern void test(void);
extern void test1(void);

#endif
```

1.2 源文件

2. 代码风格

2.1 缩进

- 【规则】TAB 缩进设置为 4 个空格
- 【规则】用空格代替 TAB 缩进

2.2 空格

- 【规则 1】运算符前后 1 个空格(一元操作符不用)

2.3 对齐

- 【规则 1】语句后的花括号不同列 且前后有 1 空格
- 【规则 2】变量初始化值=号对齐，宏定义常量 值对齐

2.3.1 if 语句

```
if(i == 0) {                    ==左右两边 1 个空格 { 前 1 个空格
    // do things
} else if(i == 2) {             } 后 1 个空格 { 前 1 个空格
    // do things
} else {                        同上
    // do things
}
```

2.3.2 循环语句

```
while(i > 1) {                  >左右两边 1 个空格
    // do things
}
for(int i = 0; i < 3; i++) {
    // do things
}
do {
    // do things
} while(i > 1);
```

2.3.3 switch 语句

```
switch(i) {
case 0:                                  case 和 switch 对齐
      break;
case 1:
      break;
case 2:
      break;
default:
      break;
}
```

2.3.4 goto 语句

```
      goto end;
end: {
         i = 1;
    }
```

2.4 长行拆分

- 【规则 1】一行代码一般不超过 80 个字符超过使用 \ 拆分符拆分(如果超出可更明确表达意思即可不分)
- 【规则 2】函数入参过多 拆分
- 【规则 3】表达式中多个运算符 拆分

```
Void test(int testbuf1, \
            int testbuf2, \
            int testbuf3, \
            int testbuf4)
{

}

If((flag1 == 1) \
   && (flag2 == 1) \
   && (flag3 == 1)) {

}
```

2.5 注释

- 【规则 1】函数声明/变量定义/程序当前行需要的注释在其上方，采用 /* 注释内容 */ 注释
- 【规则 2】当前行不重要的注释 在当前行 80 字符处 采用 // 注释内容 注释
- 【规则 3】不同类型声明或定义 通过注释 分类声明定义
- 【规则 4】如需详细说明的地方可以 加以详细注释 注意注释格式整洁

规则 1


```
/* 定义计数值变量 */
int cnt= 1 ;
```

规则 2

```
/* led 初始化 */
led_init();                    // 低电平有效
/* 蜂鸣器初始化 */
beep_init();                   // 低电平有效
/* 按键 IO 初始化 */
key_init();                    // 低电平有效
```

规则 3

```
// [include] ————————————————————————
#include <stdio.h>
// [define]—————————————————————————
#define NUM    (10)
// [global variable]————————————————————
uint8_t id = 0;
// [static variable]————————————————————
static uint8_t size = 0;
// [function declaration]———————————————
void function(void);
或采用块注释符合
/* [define]————————————————————————*/
```

3. 命名规则

采用 UNIX 风格　名称采用 小写+下划线组合。一些名称采用大写可以很好辨识的话可以采用大写 具体根据不同定义如下

3.1 变量

- 【规则】使用 名词 或 形容词+下滑线+名词 尽量使用可以轻易看懂的词
- 【规则】全局变量 前加 __g_变量名　两个下划线 + g+ 下划线 + 名称

```
/* 全局 */
int __g_num;
/* 全局常量 */
const int __cg_num=1;
/* 全局静态变量/本地变量 */
static int __sg_num=0;

int value;
int old_value;
int ad_value;
```

3.2 常量

- 【规则】全局常量 前加 __cg_变量名　两个下划线 + cg+ 下划线 + 名称
- 【规则】全局静态变量 前加 __sg_变量名　两个下划线 + sg+ 下划线 + 名称
- 【规则】函数入参/函数体内常量 无需以上标注
- 【规则】宏定义常量 全部大写 可 加下划线

```
/* 全局常量 */
const int __cg_num=1;
/* 全局静态变量/本地变量 */
static int __sg_num=0;
/* 宏定义 */
#define NUM         (10)
#define ID_NUM    (10)
```

3.3 数组

- 【规则】arr + 下划线 + 数组名 一定初始化

```
uitn8_t arr_info[10] = { 0 };
```

3.4 指针

■ 【规则】 *靠近 名称 名称采用 ptr/p +下划线+指针名

```
/* 指向字符串的指针 */
uitn8_t *p_str =NULL;
uitn8_t *ptr_str =NULL;
```

3.5 结构体

■ 【规则】 采用 typedef struct 定义 名称后+下划线+t

```
Typedef struct {
    uint8_t id;
    char name[20];
} info_t;
```

3.6 位域

■ 【规则】 采用 typedef struct 定义 名称后+下划线+bf (bit field)

```
Typedef struct {
    uint8_t id: 6;
    uint16_t sum: 10;
} sum_bf;
```

3.7 函数

【规则】 动词 + 名词

```
/* 获取数据 */
void get_data(void);
/* 写一个字节数据 */
void write_one_byte(void);
```

4. 其他