1.头文件的写法

```
#ifndef _TEST_H
#define _TEST_H
    /*your header file*/
#endif
```

在写每个头文件的时候需要在文件头部加上形式如上的代码,这么做是为了不重复包含头文件，def 后面的标识符以下划线“_”为开头，接头文件名的大写字母,最后加上“_H”（这么做是为了防止与系统中已有的.h 文件中的宏重名)。

eg:

test.h 它的头文件标识符就是:_TEST_H

2.注释

a)在每个文件头部写出如下的信息

```
/**
* Copyright (c)  你的名字
* All rights reserved.
* 程序名:
* 作者:
* 程序版本:
* 文件描述:
* 创建日期:
* 完成日期:
* 修改日期:
* 修改作者:
* 修改后的版本:
*/
```

可以根据实际情况添加、删除或修改。

b)若要为某一段代码或者函数注释,则在要注释的代码段的上部，按如下格式注释

```
/**
 * 相关注释
 */
```

c)如果用一行可注释完的话,用如下格式注释

```
/* 相关注释 */
```

d)若要为某一句代码注释,则在该句代码后用如下格式注释

```
//相关注释
```

e)注释的缩进位置需要与所注释的代码缩进位置一致

3.if switch for while 后采用建议采用如下代码规范

a)上述关键字后所接的语句即使只有一条执行语句,也要加上花括号,花括号的位置采用可采用内核编码风格，花括号的前半部分紧跟它附属的语句，或者两个花括号各占一行。

eg: 第一种风格

```
if (condition1) {
    /*code*/
} else if (condition2) {
    /*code*/
} else { //condition3
    /*code*/
}
```

eg: 第二种风格

```
if (condition1)
{
    /*code*/
}
else if (condition2)
```

```
{
    /*code*/
}
else //condition3
{
    /*code*/
}
```

b)关键字后与括号之间空一格,反括号与同行的花括号之间也要空一格 。for 的条件循环中的“;” 与下一条语句之间空一格。

eg:

```
for (i = 0; i < n; i++) {
    /*code*/
}
while (condition) {
    /*code*/
}
do {
    /*code*/
} while (condition);
```

c)如果所写代码中出现多层嵌套,或者执行代码较长,则在每个嵌套语句结束后,注释结束的是哪一个条件控制语句。

eg:

```
for ( ; ; ) {
    for ( ; ; ) {
        /*code*/
        if (condition) {
            /*code*/
        } /*end of if (condition)*/
    } /*end of for ( ; ;)*/
} /*end of for ( ; ;)*/
```

4.函数代码要注明函数返回类型和参数类型,如果没有参数要注明为void 类型。 函数代码的花括号独自成行,不要写在函数名之后。

eg:

建议写法:

```
int main(int argc, char *argv[])
{
    /*code*/
    return _TURE_;
}
```

不建议写法:

```
main () {
    /*code*/
}
```

5.双目操作符如“=”,“+=”等赋值操作符,“+”,“*”,“%“等算术操作符,“&&”等位操作符前后加空格。

6.“>”和“.”操作符前后不加空格。

7.逗号运算符后需要加空格。

eg:

```
void foo(int x, int y, int z) ;
a = (1, 2, 3);
```

8.单目操作符如“&”,“~”,“!”等前后均不加空格。

9.尽量不使用默认优先级的表达式,而使用()表示表达式的优先关系。

10.变量全部使用小写字母,宏定义全部使用大写字母,使用“_”分隔单词,为 了避免与系统内部所包含的.h 文件中的宏重名,在自己定义的宏标识符前后加上“_” 。

eg:

```
#define _TRUE_ 1
#define _FALSE_ 0
```

11.循环控制变量可以使用 i,j,k,m,n 等变量,但是其他变量需要有意义 的命名,在不影响理解的情况下可以缩写。

12.定义变量的时候需要为其初始化。

eg:

```
int a = 0;     //常量初始化为 0
double b = 0.0; //浮点型初始化为 0.0
char c = '\0'; //char 型初始化为 \0
char *pointer = NULL; //指针类型初始化为 NULL
```

13.指针定义是 “*”不要跟在类些后面,要放在变量的前面。

eg:

```
char *name = NULL;
int **pointer = NULL;
```

14.同一行最多只写一条一句。

eg:

建议写法:

```
i++;
j++;
k++;
```

不建议写法:

```
i++; j++; k++;
```

15.如果某一句代码较长用”\”分隔成多行写，如果某个函数有很多参

数,则将参数用“\”分隔成多行写。

eg:

```
foo(Type a, \
    Type b, \
    Type c, \
    Type d, \
    Type e) {
    /*code*/
}
```

17.函数之间要用两行空格隔开,函数内部中不同功能的代码片断之间用一 行空格隔开。

18.为测试方便,统一采用#ifdef DEBUG 来标记显示的测试语句。

eg:

```
#ifdef DEBUG
    printf( “infomation” );
#endif
```

如果测试语句在函数中,建议手动将#if DEBUG 以及#endif 移动到 相应的缩进位置。

eg:

```
#define DEBUG
int a = 0;
int foo (void)
{
    int a = 1;
#ifdef DEBUG
    printf( “a = %d\n” , a);
#endif
    return _TRUE_;
```

```
    }
#ifdef DEBUG
    printf( “a = %d\n” , a);
#endif
```

上述风格为建议风格，程序员可以作为参考，使用自己的代码风格，实际做项目时，请按照项目的具体代码规范来编写代码。

本文参考文献:

<<XyFTP CodeStyle>> 作者:林峰,刘洋

<<华为软件开发规范>>

建议参考：

《linux kernel coding style》

http://wenku.baidu.com/view/155bcd6548d7c1c708a14547.html?from=related&hasrec=1