**C/C++编码规范**

# 前言

编码的原则是代码结构条理清晰，注释简明扼要，变量、函数、对象、文件等的命名能明确表达它的意思，要避免模块之间、文件之间的命名冲突。

缩进规范：下级与上级之间采用4空格缩进（IDE默认的制表键和缩进为4个空格，不符合的需要调整为4个空格）。

# C语言编码规范

## 命名规范

命名原则遵循模块内不重名、模块之间不重名。命名要做到见名知意，能够通过名称就指定是那个模块实现的功能是什么。可以借助模块标识、命名空间等手段进行限定。

### 模块、包命名

模块、包名的名字由英文名称的简写小写字母组成。

示例1：

环境与设备监控系统（building automation System）为“bas”。

### 头文件、源文件命名

命名规则：系统+模块+功能全拼，即”系统\_模块\_功能.h”、”系统\_模块\_功能.c”。系统和模块为英文缩写小写字母，功能为功能名英文全拼小写。

示例1：环境与设备监控系统时间表定时执行

bas\_ttbl\_cron.h

bas\_ttbl\_cron.c

### 结构体命名

命名规则：系统+模块+对象，即”系统\_模块\_对象”。系统、模块为英文缩写小写字母，对象为英文全拼小写。

示例1:

struct bas\_ttbl\_cron\_node

{

int id; // cron ID

char \*name; // cron name

};

typedef struct bas\_ttbl\_cron\_node bas\_ttbl\_cron\_node;

### 函数命名与参数

命名规则:系统+模块+操作+目标，即“系统\_模块\_操作\_目标”。系统、模块为英文缩写小写字母，操作为英文小写全拼，目标为英文小写全拼。

示例1：

const char\* bas\_ttbl\_get\_id();

void bas\_ttble\_set\_id(const char\* id);

int bas\_ttbl\_load\_configuration(const char\* filename);

int bas\_cmn\_is\_connection\_closed();

### 变量命名

全局变量命名规则：g\_变量名称。”变量名称”为英文缩写与英文全拼的组合。

示例1：

int g\_totalCounter;

client g\_clients[10];

静态变量与常量命名：\_变量名称。”变量名称”为英文缩写与英文全拼的组合，

示例2:

static client \_client;

const int \_maxLine;

局部变量命名与其它变量命名：变量名称。”变量名称”为英文缩写与英文全拼的组合。

示例3：

float totalUsage;

int maxFileSize;

### 宏定义

命名规则：英文大写+“\_”。

示例1:

#define MAX\_LINE(c) ((c)->lines)

#define MAX\_BUF\_SIZE 128

## 控制流程规范

### IF

编码格式：if、else if、else与’{’、‘}’要分行， if、else if与”()”之间要空一个空格。“()”内的判断条件要合理的使用“()”进行限定，且避免把函数调用直接作为判定条件。

示例1：

int httpRespCode = http\_do\_post(...);

if (httpRespCode == 200)

{

printf(“ok\n”);

}

else if (httpRespCode == 400)

{

printf(“server error\n”);

}

else

{

printf(“....”);

}

示例2：

if ((size > 10 && id == 1) || (size == 0))

{

}

else

{

}

### SWITCH

编码格式：switch、case、default与’{’、‘}’要分行，switch与”()”、case与判定条件之间要空一个空格。“()”内的判断条件要合理的使用“()”进行限定，且避免把函数调用直接作为判定条件。case、default要缩进4个空格，且它们的执行语句要缩进相对的4个空格。

示例1：

switch (index)

{

case 1:

break;

case 2:

break;

default:

;

}

### WHILE

编码格式：while、‘{’、‘}’分行，while与“()”要空1个空格。“()”内条件避免有函数调用。

示例1：

while (i < 100)

{

printf();

}

示例2：

int counter = bas\_ttbl\_get\_schedules();

while (counter > 0)

{

counter--;

}

### DO WHILE

编码格式：do、‘{’、‘}’分行，while与‘}’不分行。while与“()”、‘}’之间分别空1个空格。“()”内条件避免有函数调用。

示例1：

do

{

} while (size > 0);

### FOR

编码格式：for、‘{’、‘}’分行，for与“()”之间空1个空格。“()”内避免直接出现函数调用。

示例1：

int i;

for (i = 0; i < 100; i++)

{

}

示例2：

int i;

int scheduleNum = bas\_ttbl\_get\_schedules();

for (i = 0; i < scheduleNum; i++)

{

}

### 宏函数与宏控制块

编码格式：宏参数要使用“()”进行限定，合理使用换行符‘\’。

示例1：

#define HTTP\_BUF\_MAX 1024

示例2：

#define LIST\_GET\_LENGTH(l) ((l)->length)

示例3：

#define MAX(a,b) do {\

if ((a) > (b))\

{\

return (a);

}\

else\

{\

return (b);

}\

} while (0);

# C++编码规范

## 命名规范

### 模块、包、命名空间

模块、包名、命名空间由英文名称的简写小写字母组成。

示例1：

环境与设备监控系统（building automation System），为“bas”。

### 头文件、源文件命名

一般头文件与源码文件规则：大写字母‘C’+功能的英文全拼，即“C功能.h”、“C功能.cpp”（源文件后缀名同一为.cpp）。功能为英文全拼首字母大写，独立单词首字母大写。

接口头文件规则：大写‘I’+功能的英文全拼，即“I功能.h”。

示例1：时间表

TimeTable.h

TimeTable.c

示例2：时间表计划任务

TimeTableCron.h

TimeTableCron.cpp

接口文件命名。接口头文件命名以’I’+功能的英文全拼，”I功能.h”。

示例3：

IService.h

### 类命名

类命名规则：大写字母‘C’+功能的英文全拼，即CXxxxXxxx。功能为英文全拼首字母大写，独立单词首字母大写。做到头文件与源文件一一对应，且一个头文件只定义一个类。

属性命名规则：’\_’+属性名。属性名为英文缩写与全拼组合。全拼的每一个单词首字母大写。

方法命名规则：操作+对象。

示例1：头文件（TimeTable.h）

class CTimeTable

{

public:

CTimeTable();

CTimeTable(int id, int ms);

~CTimeTable();

void setName(const std::string& name);

void setInterval(int ms);

int getInterval();

public:

int \_intervalMS; // interval in milliseconds

int \_fd; // file description

std::string \_name;

};

示例2：基类

class CBase

{

CBase();

~CBase();

};

### 结构体命名

规则：功能的英文全拼，首字母大写，独立的单词首字母大写。

示例1：

struct TimeTable

{

int id;

char \*name;

};

### 接口类命名

类命名规则：大写字母‘I’+功能的英文全拼，即IXxxxXxxx。功能为英文全拼首字母大写，独立单词首字母大写。一个接口头文件只定义一个接口类。

示例1：接口头文件（IService.h）

IService.h

# 注释与参数

## 注释

规则：注释要简明扼要，用最少语言描述最清晰思路。一眼就能看明白的地方，无需注释。函数体内尽量使用行注释，尽量避免使用块注释。接口参数注释使用块注释。

头文件与源文件头注释。简单描述本文件要实现的主要功能，以及文件的修订历史。

示例1：

/\*

@Copyright Reserved by XXXX.

This is for TCP socket. It implements methods to create connection, send

and receive data.

Created By Joe, 2011.01.22

Histories:

\*/

代码注释。注释要简明扼要，尽量用最少的文字简单描述。尽量减少不必要的注释。函数内部尽量避免使用“/\*\*/”块注释。如果函数内需要注释是尽量使用”//”注释。函数外部使用“/\*\*/”快注释。

示例2：

void xxXXX()

{

// xxxxx

}

void xx\_xx\_xx\_xx()

{

// xxxxxxx

}

函数参数注释。要说明函数作用，参数类型。

示例：

/\*

bas\_ttbl\_get\_schedule\_name get timetable’s schedule name.

arguments:

name -- I, xx name. pre-allocated memory buffer.

size -- IO, name size. input the max size and return real size..

return:

0 -- ok, !0 -- error code.

\*/

int bas\_ttbl\_get\_schedule\_name(char \*name, int \*size);

## 参数

规则：明确区分入参、出参。能使用对象参数的使用对象参数，对象参数尽量使用地址传递、引用传递，避免使用值拷贝传递。

示例1：

int xx\_set\_name(const char \*name, int length);

int xx\_get\_name(char \*name, int \*length);

示例2：

int xx\_set\_person(const person \*person);

int xx\_process\_person(person \*person);

# 其它

## 接口原则

接口类、接口函数要明确表达接口要实现的功能。接口函数与接口对象参数要分离为接口与对象。

## 内存原则

内存使用原则：谁申请谁释放。如果是外部模块提供的内容申请接口，则必须提供对应的内存释放。

尽量避免在循环内部重复申请释放同一个变量指向的内存。