C语言编码规范

（参考于软件开发代码规范（C语言）.doc）

By 吴国诚

目录

第一章 排版 2

1.1 空行 2

1.2 代码行 3

1.3 代码行内空格 4

1.4 对齐缩进 5

1.5 长行拆分 6

第二章 注释 8

2.1 通用规则 8

2.2 文件注释 8

2.3 函数注释 8

2.4 数据注释 9

2.5 代码注释 10

第三章 命名 12

3.1 通用命名规则 12

3.2 文件命名 12

3.3 类型命名 12

3.4变量命名 13

3.5 常量命名 13

3.6 函数命名 14

3.7 枚举命名 14

3.8 宏命名 14

# 排版

## 1.1 空行

【规范1.1.1】在每个函数、结构体、枚举定义结束之后都要加空行。

struct s1

{

…

};

// 空行

enum

{

…

}；

// 空行

void funtion1(…)

{  
 …

};

【规范1.1.2】在一个函数体内，逻辑密切相关的语句之间不加空行，其它地方应加空行分隔。

// 空行

while (condition)

{

statement1;

// 空行

if (condition)

{

statement2;

}

else

{

statement3;

}

// 空行

statement4;

}

【规范1.1.3】相对独立的程序块之间、变量说明之后必须加空行。

if (!is\_lock\_card\_succ)

{

... // program code

}

// 空行

GetLockPhoneInfo(&st\_lock\_phone\_info);

## 1.2 代码行

【规范1.2.1】一行代码只做一件事情，如只写一条语句。这样的代码容易阅读，并且方便于写注释。if、for、while、do等语句自占一行，执行语句不得紧跟其后。不论执行语句有多少都要加{}。这样可以防止书写失误。

|  |  |
| --- | --- |
| X ＝ a + b; y = c + d; z = e + f; | x = a + b;  y = c + d;  z = e + f; |
| if (width < height) dosomething(); | if (width < height)  {  dosomething();  } |
| for (initialization; condition; update)  dosomething();  other(); | for (initialization; condition; update)  {  dosomething();  }  // 空行  other(); |

不规范代码 规范代码

## 1.3 代码行内空格

【规范1.3.1】关键字之后要留空格。const、static等关键字之后至少要留一个空格，否则无法辨析关键字；if、for、while、switch等关键字之后应留一个空格再跟左括号‘(’，以突出关键字。

【规范1.3.2】函数名之后不要留空格，紧跟左括号‘(’，以与关键字区别。

【规范1.3.3】‘(’向后紧跟，‘)’、‘,’、‘;’向前紧跟，紧跟处不留空格。

【规范1.3.4】‘,’之后要留空格，如Function(x, y, z)。如果‘;’不是一行的结束符号，其后要留空格，如for (initialization; condition; update)。

【规范1.3.5】赋值操作符、比较操作符、算术操作符、逻辑操作符、位域操作符，如“=”、“+=” “>=”、“<=”、“+”、“\*”、“%”、“&&”、“||”、“<<”,“^”等二元操作符的前后应当加一个空格。

【规范1.3.6】一元操作符如“!”、“~”、“++”、“--”、“&”（地址运算符）等前后不加空格。

【规范1.3.7】“［］”、“.”、“->”这类操作符前后不加空格。

|  |  |
| --- | --- |
| void Func1(int x, int y, int z); | void Func1 (int x,int y,int z); |
| if (year >= 2000) | if(year>=2000) |
| if ((a>=b) && (c<=d))  if ((a >= b) && (c <= d)) | if(a>=b&&c<=d) |
| for (i = 0; i < 10; i++)  for (i=0; i<10; i++) | for(i=0;i<10;i++) |
| x = a < b ? a : b; | x=a<b?a:b; |
| i++;  int \*x = &y; | i ++;  int \* x = & y; |
| array[5] = 0;  a.Function();  b->Function(); | array [ 5 ] = 0;  a . Function();  b -> Function(); |

良好风格 不良风格

## 1.4 对齐缩进

【规范1.4.1】程序块要采用缩进风格编写。

【规范1.4.2】对齐使用TAB键，TAB键宽度设置为4个空格。

【规范1.4.3】函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格，case语句下的情况处理语句也要遵从语句缩进要求。

【规范1.4.4】程序块的分界符（如‘{’和‘}’）应各独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开始、类的定义、结构的定义、枚举的定义以及if、for、do、while、switch、case语句中的程序都要采用如上的缩进方式。

|  |  |
| --- | --- |
| for (...) {  ... // program code  } | for (...)  {  ... // program code  } |
| if (...)  {  ... // program code  } | if (...)  {  ... // program code  } |
| void example\_fun( void )  {  ... // program code  } | void example\_fun( void )  {  ... // program code  } |

不规范代码 规范代码

【规范1.4.5】预处理指令不需要缩进，总是从行首开始。即使预处理指令位于缩进代码块中，指令也应从行首开始。

|  |
| --- |
| // 良好风格：预处理指令均从行首开始  if (lopsided\_score)  {  #if DISASTER\_PENDING // Correct -- Starts at beginning of line  DropEverything();  #if NOTIFY  NotifyClient();  #endif  #endif  BackToNormal();  } |
| // 不良风格：缩进的预处理指令  if (lopsided\_score)  {  #if DISASTER\_PENDING // Wrong! The "#if" should be at beginning of line  DropEverything();  #endif // Wrong! Do not indent "#endif"  BackToNormal();  } |

## 1.5 长行拆分

【规范1.5.1】代码行最大长度宜控制在100至110个字符以内。代码行不要过长，否则眼睛看不过来，也不便于打印。

【规范1.5.2】较长的语句（>110字符）要分成多行书写；长表达式要在低优先级操作符处拆分成新行，操作符放在新行之首（以便突出操作符）。拆分出的新行要进行适当的缩进，使排版整齐，语句可读。

【规范1.5.3】循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分. 长表达式要在低优先级操作符处划分新行，操作符放在新行之首。

【规范1.5.4】若函数或过程中的参数较长，则要进行适当的划分。

|  |
| --- |
| if ((very\_longer\_variable1 >= very\_longer\_variable12)  && (very\_longer\_variable3 <= very\_longer\_variable14)  && (very\_longer\_variable5 <= very\_longer\_variable16))  {  dosomething();  } |
| virtual CMatrix CMultiplyMatrix (CMatrix leftMatrix,  CMatrix rightMatrix); |
| for (very\_longer\_initialization;  very\_longer\_condition;  very\_longer\_update)  {  dosomething();  } |
| report\_or\_not\_flag = ((taskno < MAX\_ACT\_TASK\_NUMBER)  && (n7stat\_stat\_item\_valid (stat\_item))  && (act\_task\_table[taskno].result\_data != 0)); |
| n7stat\_str\_compare((BYTE \*) & stat\_object,  (BYTE \*) & (act\_task\_table[taskno].stat\_object),  sizeof (\_STAT\_OBJECT)); |

# 注释

## 2.1 通用规则

【规范2.1.1】一般情况，需要保证程序有一定的注释。必须保证关键的函数、流程、类型定义、变量等有相应注释说明。

【规范2.1.2】注释应当准确、易懂，防止注释有二义性。

【规范2.1.3】避免在注释中使用缩写，特别是非常用缩写。

【规范2.1.4】通过对函数或过程、变量、结构等正确的命名以及合理地组织代码的结构，使代码成为自注释的。

【规范2.1.5】注释格式尽量统一。建议使用//进行注释，多行注释可使用“/\* …… \*/”。

## 2.2 文件注释

【规范2.2.1】源文件（包含.h头文件、.c源文件及各种脚本文件等）头部应进行注释，应列出：版权说明、文件名、文件目的/功能，作者、创建日期等。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Copyright (C) 2015-2016, XXX Co. Ltd.

\*\* All rights reserved.

\*\*

\*\* FileName: // 文件名称

\*\* Description: // 文件描述

\*\* Author: // 作者

\*\* Date: // 创建时间

\*\* Others: // 其它说明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## 2.3 函数注释

【规范2.3.1】函数头部应进行注释，需要列出函数的功能、参数、返回值等。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Function: // 函数名称

// Description: // 函数功能描述

// Param: // 参数说明，包括参数的作用、取值范围等

// Return: // 函数返回值说明

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## 2.4 数据注释

【规范2.4.1】对于所有有物理含义的变量、常量，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。变量、常量、宏的注释应放在其上方相邻位置或右方。

|  |
| --- |
| // active statistic task number  #define MAX\_ACT\_TASK\_NUMBER 1000  #define MAX\_ACT\_TASK\_NUMBER 1000 // active statistic task number |

【规范2.4.2】数据结构声明(包括结构体、枚举、类等)，如果其命名不是充分自注释的，必须加以注释。对数据结构的注释应放在其上方相邻位置；对结构中每个域的注释放在该域的右方。

|  |
| --- |
| // sccp interface with sccp user primitive message name  enum SCCP\_USER\_PRIMITIVE  {  N\_UNITDATA\_IND, // sccp notify sccp user unit data come  N\_NOTICE\_IND, /\* sccp notify user the No.7 network can not  transmission this message\*/  N\_UNITDATA\_REQ, // sccp user's unit data transmission request  }; |

【规范2.4.3】全局变量必须有注释，包括对其功能、取值、及其他注意事项等的说明。

|  |
| --- |
| //标志是否通过锁卡流程；TURE = 通过锁卡流程，FALSE = 锁卡流程失败  PUBLIC BOOLEAN g\_isLockCardPass = FALSE; |

## 2.5 代码注释

【规范2.5.1】边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除！

【规范2.5.2】如果代码本来就是清楚的，则不必加注释。

|  |
| --- |
| i++; // i 加 1，多余的注释 |

【规范2.5.3】在代码的功能、意图层次上进行注释，提供有用、额外的信息。

|  |  |
| --- | --- |
| // if receive\_flag is TRUE  if (receive\_flag) | // if mtp receive a message from links  if (receive\_flag) |

无用注释 有用注释

【规范2.5.4】注释应与其描述的代码相邻。对语句块的注释必须放在语句块上方；对单条语句、变量定义的注释可以放在上方或右方（建议放在右方）；注释不可放在下方。

【规范2.5.5】如果注释放在上方，则将注释与其上面的代码用空行隔开。

|  |  |
| --- | --- |
| // code one comments  program code one  // code two comments  program code two | //code one comments  program code one    // code two comments  program code two |

过于紧凑 良好写法

【规范2.5.6】避免在一行代码或表达式的中间插入注释。

【规范2.5.7】对于switch语句下的case语句，如果因为特殊情况需要处理完一个case后进入下一个case处理，必须在该case语句处理完、下一个case语句前加上明确的注释。

【规范2.5.8】注释与所描述内容进行同样的缩排。

|  |  |
| --- | --- |
| void example\_fun( void )  {  //code one comments  CodeBlock One  //code two comments  CodeBlock Two  } | void example\_fun( void )  {  // code one comments  CodeBlock One  // code two comments  CodeBlock Two  } |

不好的注释缩排 良好的注释缩排

# 命名

## 3.1 通用命名规则

【规范3.1.1】标识符的命名要清晰明了，有明确含义；命名应具有描述性；一般而言，类型和变量应是名词，函数应是“命令性”动词；

|  |
| --- |
| int counter; //计数器——名词  NET\_ICON\_TYPE icon\_type; // NET\_ICON\_TYPE 网络图标类型——名词  //设置IDLE页面网络图标类型——“命令性”动词  NET\_ICON\_TYPE StarLib\_GetIdleNetIconType(); |

【规范3.1.2】命名应使用使用完整的单词或大家可以理解的缩写，避免使人产生误解

## 3.2 文件命名

【规则3.2.1】文件命名采用驼峰式，小写字母开头。

|  |
| --- |
| 不良的文件命名：  mmieventmanager.h（全小写） |
| 良好的文件命名：  gameConfig.c  heroWeapon.h |

## 3.3 类型命名

【规范3.3.1】结构体（struct）、枚举（enum）类型名遵循如下规则：开头大写，采取驼峰式

struct AudioCoder

{

…

};

enum StarType

{

…

};

【规范3.3.2】类型定义（typedef）类型开头带下划线且全大写，后面的采用开头大写的驼峰式

typedef strcut \_GAME\_CONFIG

{

…  
}GameConfig;

## 3.4变量命名

【规范3.4.1】包括局部变量、全局变量、参数变量、成员变量，变量名一律开头小写，采用驼峰式

char \*namePtr;

int lenOfArray;

【规范3.4.2】对于变量命名，禁止使用单个字符（如i、j、k等）。i、j、k等仅能用作局部循环变量。

|  |
| --- |
| 单个字母唯一可使用的场合：  for (i = 0; i < max; i++)  {  …  } |

## 3.5 常量命名

【规范3.5.1】常量名全部字母大写。

|  |
| --- |
| const float PI = 3.14;  const int VAL\_MIN = 1; |

## 3.6 函数命名

【规范3.6.1】函数名中每个单词首字母小写，采用驼峰式。

## 3.7 枚举命名

【规范3.7.1】枚举值全部大写，单词间使用下划线相连

|  |
| --- |
| typedef enum  {  STAR\_STARTUP\_ANIM\_G3\_LOGO = 0x00, //移动G3 LOGO,默认  STAR\_STARTUP\_ANIM\_CHINA\_TIETONG = 0x01, //中国铁通  STAR\_STARTUP\_ANIM\_BEIJING\_MOBILE = 0x02, //北京移动  STAR\_STARTUP\_ANIM\_STARNET = 0x03, //星网锐捷  STAR\_STARTUP\_ANIMA\_MAX,  }STAR\_STARTUP\_ANIM\_TYPE\_E; |

## 3.8 宏命名

【规范3.8.1】宏命名全部大写，单词间使用下划线相连。

|  |
| --- |
| #define STAR\_DM\_DES\_NUM\_MAX\_LEN (20) //DM短信目的地址最大长度  #define STAR\_REAL\_LCD\_CONTRAST\_DEFAULT (4) //液晶默认对比度 |