

安服优物联网（AFUIOT）开放平台

C语言编程规范

Copyright  ©2019 AFUIOT.COM，版权所有：深圳市安服优智能互联科技有限公司。本文档权利使用CC-By 3.0许可证授予，该许可证鼓励您共享使用此文档，详细信息请查看：<https://creativecommons.org/licenses/by/3.0/>

|  |  |  |
| --- | --- | --- |
| 版本 | 修订 | 日期 |
| V1.0 | 宋阳 | 2019.2.17 |

# 背景

本文档是安服物联网开放平台C语言编程的规范文件，AFUIOT开放平台项目的所有C语言代码均采用此规范编写，本规范基于谷歌开源项目C++风格指南编写，与Google C++ Style Guide兼容，可视为其C 语言的子集，汇编了C语言代码风格的常用要求，建议使用AFUIOT开放平台的C语言代码遵循此文档的要求，同时需详细查阅Google C++ Style Guide。

本文档包含于AFUIOT开放平台组件内，并包含有文件模版以及使用本规范编写的.c与.h文件用例，还提供了Google C++ Style Guide的中英文版本，亦可查询以下链接：

<https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/>

<https://github.com/google/styleguide>

# 头文件

* 1. 通常每一个.c文件都需要有对应的.h文件；
  2. 使用define保护头文件，格式为<PROJECT>\_<FILE>\_H\_；
  3. 兼容C/C++的混合编程，使用下列宏来实现：

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*\*\*\* some declaration or so \*\*\*\*\*/

#ifdef \_\_cplusplus

  }

#endif /\* end of \_\_cplusplus \*/

* 1. 头文件应该自给自足，即包含此文件时不需要再含额外的头文件来保证编译正确；
  2. 应该使用include包含头文件，不要在.c文件里extern引入函数或者全局变量；
  3. 不要在头文件中定义普通函数，如果定义内联函数则内联函数通常不要超过10行。
  4. 避免没有使用的include，只include必要的，include的顺序为：

1. 系统头文件
2. 其它库的头文件
3. 项目头文件

# 作用域

* 1. 遵循尽可能小的作用域原则，减少接口数量，如一个全局变量只在单个函数内使用就在这个函数内用static定义成局部静态变量，如只在单个.c文件内使用则在此文件中定义成全局静态变量，如函数只在一个.c文件中使用则把此函数用static定义成局部函数；
  2. 函数内的局部变量需定义在函数开头，不能定义在语句中。

# 函数

* 1. 参数顺序

函数的参数顺序为: 输入参数在先, 后跟输出参数.

说明：

C/C++ 中的函数参数或者是函数的输入, 或者是函数的输出, 或兼而有之. 输入参数通常是值参或 const 引用, 输出参数或输入/输出参数则一般为非 const 指针. 在排列参数顺序时, 将所有的输入参数置于输出参数之前. 特别要注意, 在加入新参数时不要因为它们是新参数就置于参数列表最后, 而是仍然要按照前述的规则, 即将新的输入参数也置于输出参数之前。

* 1. 声明与定义函数时一定要定义参数名，不要只定义类型，如：

void function(char \*, int); //不要使用

* 1. 编写简短的函数，通常一个函数不要超过40行，但如果函数内部是紧密关联的，且无需拆分成不同的模块供调用，则可以编写较长的函数。

# 命名约定

最重要的一致性规则是命名管理. 命名的风格能让我们在不需要去查找类型声明的条件下快速地了解某个名字代表的含义: 类型, 变量, 函数, 常量, 宏, 等等, 甚至. 我们大脑中的模式匹配引擎非常依赖这些命名规则。

命名规则具有一定随意性, 但相比按个人喜好命名, 一致性更重要, 所以无论你认为它们是否重要, 规则总归是规则，即要严格遵守规则。

* 1. 函数命名, 变量命名, 文件命名要有描述性; 少用缩写，类型和变量应该是名词，函数名，带参数的宏可以用“命令性”动词，对于广泛使用的特殊名词缩写应保持缩写单词大写，如TCP、VIP、AFU；
  2. 一些特定的广为人知的缩写是允许的，例如i表示迭代变量；
  3. 文件命名全部小写，以下划线连接；
  4. 类型命名每个单词首字母均大写，不包含下划线，不同的类型以不同的尾辍结束

结构体类型以Def结尾;

函数指针以Fun结尾;

枚举类型以Enum结尾；

除此之外需减少重定义类型的数量，尽量使用标准类型，优先使用C99标准的uint8\_t一类有符号与位宽的类型；

* 1. 变量命名，全部使用小写，以下划线连接，包括普通变量、结构体成员、函数参数，在此基础上：

全局变量：以g\_为前辍；

静态变量：以static定义的全局静态变量及局部静态变量以gs\_为前辍；

5.6 常量命名：大小写混合，单词首字母需大写，并以k开头，如kDeviceID；

* 1. 函数命名，以大小写混合，单词首字母需大写；
  2. 枚举命名，同常量;
  3. 宏命令，全部大写，以下划线连接。
  4. 命名分类表：

|  |  |  |
| --- | --- | --- |
| 命名规则 | 命名目标 | 附加说明 |
| 全小写，以下划线连接 | 文件、普通变量、函数参数、结构体成员、全局变量、静态变量 | 全局变量以g\_开头，静态变量以gs\_开头 |
| 大小写混合，单词首字母大小，无下划线 | 普通类型、结构体类型、枚举类型、函数指针类型、函数、常量、枚举 | 结构体类型以Def结尾，枚举类型以Enum结尾，函数指针以Fun结尾，常量与枚举以小写字母k开头 |
| 全大小，以下划线连接 | 宏、带参数的宏 |  |

# 注释

* 1. 注释风格，大括号外使用/\*\*/，大括号内可使用//注释，/\* 与// 后总是有一个空格；
  2. 文件注释，每一个文件开头应该有版权公告，文件内需划分板块，不同的代码写入对应的板块中，文件注释与板块划分可参考文件模版；
  3. 函数注释，使用JAVADOC格式；
  4. 变量注释，一般情况下，变量名就是足够的注释，但是全局变量与常量需要添加详细的注释，特别是在变量需要特殊的赋值或调用方法时；
  5. 函数内注释，比较隐晦的地方、需要添加注释；
  6. TODO注释，临时或未完成的工作需使用大写TODO注释

# 格式

* 1. 每一行的长度不要超过80行，请使用编辑器的行长度标注功能；
  2. 使用UTF-8 NOBOM编码，如果引入的代码是非UTF-8 NOBOM的，请先转换格式后再编辑；
  3. 只使用空格，每次缩进2个空格，将编辑器的tabs设置成转换为空格；
  4. 函数的声明与定义，返回类型与函数名放在同一行，合适的话参数也放在同一行，如果一行放不下，则参数可以与第一参数对齐；
  5. 函数调用，尽量放在同一行；
  6. 不要在圆括号中添加空格，如果语句中哪一分支使用了大括号，其他部分也必须使用；
  7. 空循环体应使用{}或continue，而不是一个简单的分号；
  8. 函数返回值不要使用圆括号；
  9. 预处理指令不要缩进；
  10. 水平留白，水平留白的使用根据在代码中的位置决定. 永远不要在行尾添加没意义的留白，详细说明见文件模版；
  11. 垂直留白，垂直留白越少越好，不到万不得已，不要使用空行；
  12. 充分使用编辑器的beautify功能，Google提供了eclipse的beautify配置文件，有些编辑器也支持Google Style的自动beautify功能。

# 其它说明

* 1. 现有不统一代码，对于现有不符合既定编程风格的代码可以网开一面，但是编写新的文件时需使用本规范；
  2. 充分利用编辑器，选择一款好的编辑器，除了可以提升编码的便利性，也可以在编辑代码时或使用beautify功能后使代码自动符合本规范；
  3. 查看文件模版中使用本规范编程的实例。