嵌入式C语言编码规范（0.1版）

[0规范原则 2](#_Toc466059518)

[1目录结构 2](#_Toc466059519)

[2头文件 3](#_Toc466059520)

[2.1文件命名 3](#_Toc466059521)

[2.2版权声明 3](#_Toc466059522)

[2.3引用保护 3](#_Toc466059523)

[2.4头文件依赖 4](#_Toc466059524)

[2.5函数定义 4](#_Toc466059525)

[3源文件 5](#_Toc466059526)

[3.1文件命名 5](#_Toc466059527)

[3.2版权声明 5](#_Toc466059528)

[3.3头文件依赖 6](#_Toc466059529)

[3.4函数定义 6](#_Toc466059530)

[4命名风格 6](#_Toc466059531)

[4.1原则 7](#_Toc466059532)

[4.2缩写 7](#_Toc466059533)

[4.3类型命名 7](#_Toc466059534)

[4.4变量命名 8](#_Toc466059535)

[4.5常量命名 8](#_Toc466059536)

[4.6函数命名 8](#_Toc466059537)

[4.7枚举命名 9](#_Toc466059538)

[4.8宏命名 9](#_Toc466059539)

[5注释方式 9](#_Toc466059540)

[5.1注释风格 9](#_Toc466059541)

[5.2函数注释 10](#_Toc466059542)

[5.3变量注释 10](#_Toc466059543)

[5.4编译宏注释 11](#_Toc466059544)

[6代码格式 11](#_Toc466059545)

[6.1缩进 11](#_Toc466059546)

[6.2空行 12](#_Toc466059547)

[6.3变量声明 12](#_Toc466059548)

[6.4变量定义 12](#_Toc466059549)

[6.5结构体声明 13](#_Toc466059550)

[6.6结构体定义 13](#_Toc466059551)

[6.7函数声明 13](#_Toc466059552)

[6.8函数定义 14](#_Toc466059553)

[6.9函数调用 15](#_Toc466059554)

[6.10条件语句 15](#_Toc466059555)

[6.11循环语句 16](#_Toc466059556)

0规范原则

朴实，避免晦涩语法

严谨，逻辑反复思考

简约，命名简洁、代码精炼

性能，通过算法、编译器、硬件进行优化

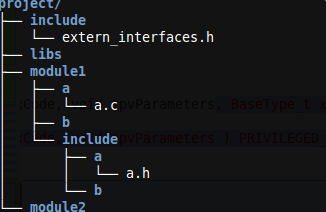
1目录结构

**原则**：

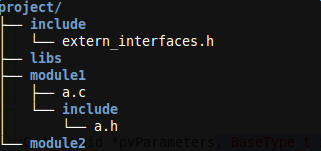
整个工程按照功能模块划分子目录，每个子目录再划分头文件和源文件目录，以便架构清晰、易懂。

**示例**：

在比较复杂的项目时，建议目录结构如下：



在项目相对简单时，建议目录结构如下：



2头文件

2.1文件命名

**原则**：

头文件命名能准确描述文件所包含的模块内容，达到通俗、易懂的目的。

**示例：**

略

2.2版权声明

**原则**：

项目内的所有头文件（除引入的第三方库）版权声明一致，并且在文件内顶格注释。

**示例：**

如YunOS版权声明如下：



2.3引用保护

**原则**：

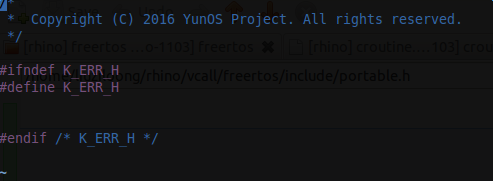
项目内的所有头文件采用#define宏防止多重包含，命名格式为：

* \*\_FILENAME\_H

注：为了保证宏定义的惟一性，建议“\*”命名为头文件目录名、路径名或有意义的其他符号（如“k”： kernel）。

**示例：**

如下图，在版权声明后空一行，且在“#endif”后空一行且添加注释（注释内容两边各留一个空格且采用“/\* \*/”格式）。



2.4头文件依赖

**原则：**

在头文件中因定义数据结构或声明函数而需要引入其他头文件中的结构体定义时，采用extern的方式引入，避免采用include头文件的方式。

**示例：**

略

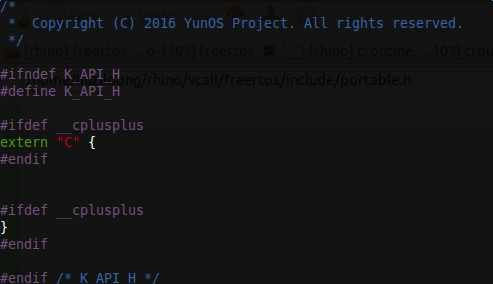
2.5函数定义

**原则：**

由于C++编译器在编译时与C编译器命名不同，为了保证C++代码能够正确调用C接口，建议在C语言接口声明时采用extern的方式指示编译器按C语言去编译。

**示例：**

如下图，在extern的声明代码上下各添加一行空行。



3源文件

3.1文件命名

**原则**：

源文件命名能准确描述文件所包含的模块内容，达到通俗、易懂的目的。

**示例：**

略

3.2版权声明

**原则**：

项目内的所有头文件（除引入的第三方库）版权声明一致，并且在文件内顶格注释。

**示例：**

如YunOS版权声明如下：



3.3头文件依赖

**原则：**

在源文件中引入头文件时采用如下顺序：**C库、C++库、其他第三方库、项目内头文件**，同一类型的头文件采用字母顺序排列，避免引入冗余头文件。

**示例：**

略

3.4函数定义

**原则：**

源文件中函数的定义顺序严格按照头文件中声明的顺序排列，且内部函数添加static关键字，比较短的函数还需添加inline关键字。对外函数必须进行参数检查，内部函数不做检查参数要求。返回值必须类型匹配。

**示例：**

略

4命名风格

4.1原则

统一，相同含义的单词、命名方式项目内统一。

准确，采用计算机领域常用单词。

风格，采用字符加下划线的方式（不使用驼峰风格）

4.2缩写

**原则：**

建议采用计算机领域常用缩写以便清晰表达含义，相关缩写可参考<http://www.abbreviations.com/abbreviation/DISABLED>。

**示例：**

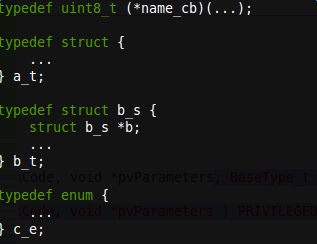
略

4.3类型命名

**原则：**

数据结构类型以“\_t”结尾，枚举变量类型以“\_e”结尾，回调函数类型以“\_cb”结尾；或所有类型以“\_t”作为后缀。字符均小写。

**示例：**



4.4变量命名

**原则：**

全局变量以“g\_”作为前缀，内部变量尽量简洁，每个变量的字段不超过5个，字符均小写。

**示例：**

略

4.5常量命名

**原则：**

常量命名各字段必须大写，且字段数不超过5个。

**示例：**

略

4.6函数命名

**原则：**

对外的函数建议采用统一前缀，如“yunos\_”,源文件内部函数建议以“\_”作为前缀，且命名字段数不超过5个，字符均小写。

**示例：**

略

4.7枚举命名

**原则：**

枚举命名各字段必须大写，且字段数不超过5个。

**示例：**

略

4.8宏命名

**原则：**

宏命名各字段必须大写，且字段数不超过5个。

**示例：**

略

5注释方式

5.1注释风格

**原则：**

统一采用“/\*\*/”方式注释，且注释内容左右各留一个空格，如“/\* message \*/”。

**示例：**

略

5.2函数注释

**原则：**

对外接口按doxgen要求的格式书写，且注释和函数定义间不留空行；对内接口在需要时添加注释，注释位于定义前。

**示例：**

略

5.3变量注释

**原则：**

注释在变量定义后，在仅有一个需要注释的变量时与分号之间只留一个空格；若多行变量定义均需注释，则最长变量定义留一个空格，其他注释与其对齐。

**示例：**



5.4编译宏注释

**原则：**

注释在宏定义后，在仅有一个需要注释的宏时与分号之间只留一个空格；若多行宏定义均需注释，则最长宏定义后留一个空格，其他注释与其对齐。

**示例：**



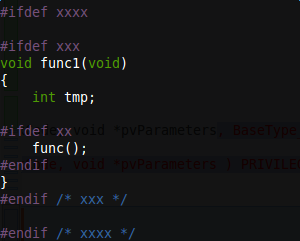
6代码格式

6.1缩进

**原则：**

所有的缩进必须使用**空格**（除makefile外不允许使用tab），且以四个空格为单位进行缩进。嵌套的编译条件判断顶格写，且需要在“#endif”后的注释中备注属于哪个判断条件。

**示例：**



6.2空行

**原则：**

在所有需要空行的地方仅空**一行**，如版权声明与代码之间，不同类型的头文件引用之间，函数定义之间等。

**示例：**

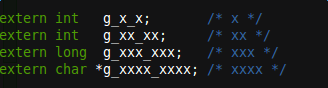
略

6.3变量声明

**原则：**

全局变量的声明必须放在相关头文件中，且单个声明的类型和变量之间留一个空格，多个声明时按最长的留一个空格对齐。

**示例：**



6.4变量定义

**原则：**

全局变量的定义必须放在相关的源文件中，且在头文件和函数定义之间；局部变量的定义必须放在函数的最开始处。

**示例：**

略

6.5结构体声明

**原则：**

在头文件中引用其他项目内头文件的结构体定义时采用extern进行声明，不引入相关头文件。

**示例：**

略

6.6结构体定义

**原则：**

结构体定义放在相关头文件中，并且在函数声明前，结构体定义前后需要各空一行。

**示例：**

略

6.7函数声明

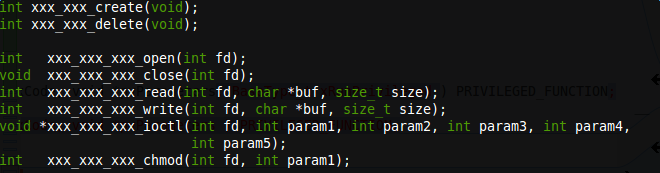
**原则：**

对外函数的声明必须放在头文件中，且在头文件中按照关联性排序；当每行超过**80列**时需将参数移到下一行且与上一行参数首字符对齐；函数内紧邻左右大括号的参数不需要空格，但参数之间需要留一个空格；仅一个函数时，返回类型与函数定义之间空一格，当多个关联函数时，返回最长的类型与函数定义之间空一格，其他关联函数首字符与其对齐。

对内声明的参数除放在源文件中外，其他与对外声明的函数保持一致。

**示例：**

命名顺序、空格、空行、参数换行请参考下图。

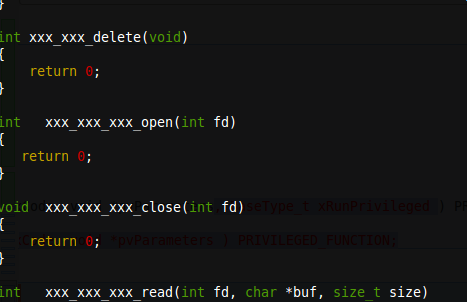


6.8函数定义

**原则：**

函数的定义原则上在源文件中实现，且其定义的顺序与声明保持一致；在函数定义时，需要将大括号另起一行，且代码逻辑紧邻大括号后一行；函数之前留一空行。

**示例：**

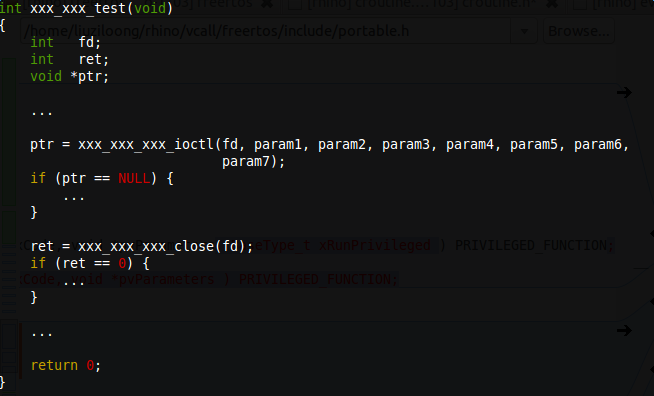


6.9函数调用

**原则：**

函数调用风格与定义基本相同，在超过80列时按照相同的规范处理；需要对函数调用的返回值进行判断，且判断语句与上一条语句之间不留空行。

**示例：**

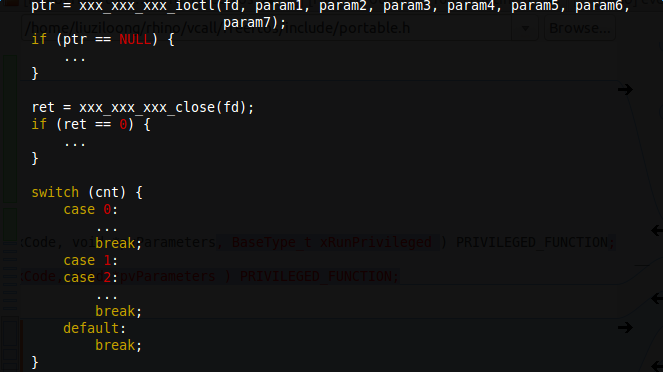


6.10条件语句

**原则：**

条件判断必须明确，如“if (var)”形式应该写为“if (var > 0)”；指针是否为空时必须与NULL进行对比（不建议用0），整型返回值可通过与1/0进行对比。

**示例：**



6.11循环语句

**原则：**

风格与条件判断语句类似。

**示例：**

