**Orca编码规范**

修改记录

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 版本号 | 修订内容 | 修订人 |
| 2018.01.02 | 0.1 | 初始化版本 | 顾伟 |

## 1.说明

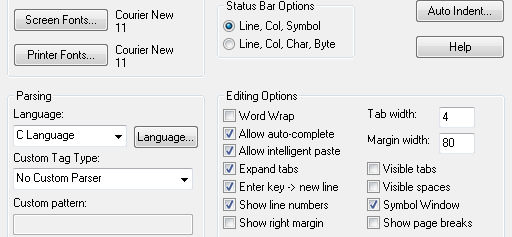
1.1 适用范围

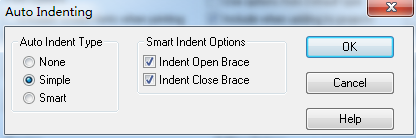
本文档针对的是C程序的编码规范

1.2 编辑器

为了统一风格，推荐使用source insight来编写代码

1.2.1 source insight配置





字体：等宽字体Courier New

Expand tabs：启用

Tab width：4

## 排版

2.1 缩进

说明

所有缩进使用tab键，并且配置tab转换为4个空格

2.2 空行

说明

空行用于区分逻辑上不同的程序块，适当加入空行可以增加整体的美观

实例

|  |
| --- |
| ISP\_DEV isp\_dev = 0;  ISP\_EXPOSURE\_ATTR\_S exp\_attr = {0};  if (HI\_MPI\_ISP\_GetExposureAttr(isp\_dev, &exp\_attr) != 0)  {  ERROR("HI\_MPI\_ISP\_GetExposureAttr fail...\n");  return FAIL;  } |

2.3 空格

说明

二元操作符的前后应当加空格

一元操作符前后不加空格

if、 for、 while等关键字之后应留一个空格再跟左括号‘(’，以突出关键字

实例

|  |
| --- |
| duty += step;  i++;  if (fp != NULL)  {  fclose(fp);  fp = NULL;  } |

2.4 对齐

说明

分界符‘{’和‘}’应独占一行并且位于同一列

{}之内的代码块在‘{’右边缩进一个单位处左对齐

函数参数较长，要进行适当的划分

实例

|  |
| --- |
| if (0 == flag)  {  if (conv(param) != OK)  {  ERROR("conv fail...\n");  return FAIL;  }  }  else  {  ERROR("flag %d is invalid...\n", flag);  return FAIL;  } |

2.5 长度限制

说明

一行代码长度限制在120个字符以内

2.6 注释

说明

单行注释使用：//

多行注释使用：/\* \*/

## 命名

3.1 文件名命令

说明

英文字符使用小写字母，不同单词之间使用”\_”作为连接符

头文件类型为.h，源文件类型为.c

实例

|  |
| --- |
| vpif\_capture.h  vpif\_capture.c |

3.2 变量命名

说明

英文字符使用小写字母，不同单词之间使用”\_”作为连接符

全局变量加入前缀”g\_”

指针变量加入前缀”p\_”，并且\*和前面的类型有一个空格，和后面的变量名结合

实例

|  |
| --- |
| Static int g\_count = 0; |

3.3 函数名

说明

英文字符使用小写字母，不同单词之间使用”\_”作为连接符

实例

|  |
| --- |
| Char \*p\_head = NULL  static int init\_default\_alg\_param(void \*p\_alg\_context) |

3.4 自定义类型

说明

不建议重新定义int/float等内置类型

自定义的结构体带”\_S”后缀

自定义的枚举带”\_E”后缀，枚举的0值分配给一个无效的类型

函数指针带”pf\_”前缀

实例

|  |
| --- |
| typedef enum hw\_check\_mode\_e  {  HW\_CHECK\_ERROR = 0,  HW\_CHECK\_LOOP,  HW\_CHECK\_RANDOM,  HW\_CHECK\_NUM,  } HW\_CHECK\_MODE\_E;  typedef struct ae\_isp\_param\_s  {  int t;  int hist\_error;  } AE\_ISP\_PARAM\_S;  typedef int (\*pf\_gpio\_init)(void \*); |

## 变量的使用

4.1 使用规则

说明

一行代码定义一个变量，不能出现”int index, count;”这样的代码

变量定义的时候都要赋初始值

尽量少用全局变量，只在一个文件中使用的全局变量使用static修饰符

实例

|  |
| --- |
| int ret = 0;  char sz\_type[64] = {0}; |

## 函数的使用

5.1 使用规则

说明

作用域仅在单个定义文件内部的函数，一律使用static进行修饰

参数的参数不要超过5，超过5的情况使用结构体传递参数

函数的实现一般不要超过200行

函数一般都需要有返回值，对于返回成功失败的函数，0表示成功，负数表示失败

函数内部要对数据的有效性进行检查，比如空指针

非特殊情况，函数内部不要使用static局部变量

基础函数模块设计的时候力求只完成一件事情

函数调用传递参数的时候类型需要匹配，不要使用隐式类型转换

实例

|  |
| --- |
| static int init\_custom\_cfg\_param(ALG\_CONTEXT\_S \*p\_alg\_context)  {  if (NULL == p\_alg\_context)  {  ERROR("input param is error...\n");  return FAIL;  }  ...  return OK;  } |

## 指针的使用

6.1 使用规则

说明

指针变量定义的时候需要带有”p\_”前缀以标示为一个指针变量

指针变量在定义的时候就需要赋初始值，防止野指针

Malloc分配的指针变量在free之后需要赋值为NULL，防止野指针

不允许使用三重指针，非必要情况不要使用两重指针

实例

|  |
| --- |
| Void \*p\_buff = NULL;  If (p\_buff != NULL)  {  Free(p\_buff);  p\_buff = NULL;  } |

## 内存的分配和释放

7.1 使用规则

说明

每写一个malloc的时候，所以的退出路径上都写好free

尽量一次性分配好内存，避免频繁的malloc和free

## 杂项

8.1 处理每个编译的warning

说明

编译时打开-Wall选项，处理每个编译警告

8.2 goto语句的使用

说明

Goto语句只能有一种用途就是异常情况退出，其他的情况尽量不要使用goto

实例

|  |
| --- |
| memset(sz\_line, 0, sizeof(sz\_line));  fgets(sz\_line, ALG\_FILE\_LINE, fp);  p\_head = sz\_line;  new\_alg\_param.ae\_speed[0] = atoi(p\_head);  if ((p\_head = strstr(p\_head, ",")) == NULL)  {  DEBUG("parse end of file\n");  goto EXIT;  } |

8.3 ==语句的使用

说明

使用==判断变量和常量是否相当的时候常量写在前面，变量写在后面，防止==写成了=，变成了赋值操作

实例

|  |
| --- |
| If (1 == val)  {  ...  } |

8.4 不依赖运算符的优先级，全部使用括号

说明

不依赖运算符的优先级，全部使用括号

实例

|  |
| --- |
| val = ((a + b) \* c) / (e + d) // 只是演示代码，变量命名不可取 |

8.5 ...