# 嵌入式C语言之-

# 标识符的命名规范

讲师：叶大鹏



助力你成为优秀的电子工程师！

# 标识符的命名规范

- **标识符命名规则：**

1. 可以由数字、字母、下划线_组成；
2. 不能以数字开头；
3. 不能是关键字；
4. 区分大小写。

# 标识符的命名规范

- **常用标识符命名风格：**

**1. unix like风格：**

单词用小写字母，每个单词直接用下划线‘_’分割，例如：text_mutex, kernerl_text_address。

**2. 小驼峰命名法：**

当标识符是一个单词的时候，首字母小写，例如：name。

当标识符由多个单词组成的时候，第一个单词首字母小写，其他单词首字母大写，例如firstName。

**3. 大驼峰命名法：**

当标识符是一个单词的时候，首字母大写，例如：Student。

当标识符由多个单词组成的时候，每个单词的首字母大写，例如：GoodStudent。

**4. 匈牙利命名法：**

用这种方法命名的变量主要包括三个部分：基本类型、一个或更多的前缀，一个限定词，例如chGrade，其中ch表示char数据类型，这种命名法使用起来比较麻烦，不太建议使用。

# 鸿蒙liteos开源代码

```
STATIC VOID OsTraceSetFrame(TraceEventFrame *frame, UINT32 eventType, UINTPTR identity, const UINTPTR
*params,
    UINT16 paramCount)
{
    INT32 i;
    UINT32 intSave;

    (VOID)memset_s(frame, sizeof(TraceEventFrame), 0, sizeof(TraceEventFrame));

    if (paramCount > LOSCFG_TRACE_FRAME_MAX_PARAMS)
    {
        paramCount = LOSCFG_TRACE_FRAME_MAX_PARAMS;
    }

    TRACE_LOCK(intSave);
    frame->curTask   = OsTraceGetMaskTid(LOS_CurTaskIDGet());
    frame->curPid    = LOS_GetCurrProcessID();
    frame->identity  = identity;
    frame->curTime   = HalClockGetCycles();
    frame->eventType = eventType;
    TRACE_UNLOCK(intSave);

    for (i = 0; i < paramCount; i++) {
        frame->params[i] = params[i];
    }
}
```

# 函数命名规则

- **建议使用大驼峰命名法:**

```
STATIC VOID OsTraceSetFrame(TraceEventFrame *frame, UINT32 eventType, UINTPTR identity, const UINTPTR *params,
    UINT16 paramCount)
{
        ...
}

UINT32 GetFatSectorsPerBlock(VOID)
{
        ...
}
```

# 局部变量命名规则

- **建议使用小驼峰命名法:**

```
STATIC VOID OsTraceSetFrame(TraceEventFrame *frame, UINT32 eventType, UINTPTR identity, const UINTPTR
*params,
    UINT16 paramCount)
{
    INT32 i;
    UINT32 intSave;

    for (i = 0; i < paramCount; i++) {
        frame->params[i] = params[i];
    }
}
```

**注：允许定义i、j、k作为局部循环变量。**

# 全局变量和静态局部变量命名规则

- **全局变量在小驼峰的基础上增加 “g_” 前缀:**

```
static bool g_enableTrace = FALSE;
static bool g_traceMask = TRACE_DEFAULT_MASK;
```

- **静态局部变量在小驼峰的基础上增加 “s_” 前缀:**

```
static int pm25_power_control(void)
{
    static uint32_t  s_minuteCount = 0;
    static uint32_t  s_openWaitTime = HIKE_PM_OPEN_WAIT_TIME_MS;
}
```

# 宏定义命名规则

- **建议采用全大写的英文字母，单词之间加下划线 '_' 的方式命名:**

```
#define TRACE_LOCK(state)          (state) = LOS_IntLock()
#define TRACE_UNLOCK(state)        LOS_IntRestore(state)

STATIC VOID OsTraceSetFrame(TraceEventFrame *frame, UINT32 eventType, UINTPTR identity, const UINTPTR
*params,
    UINT16 paramCount)
{

    if (paramCount > LOSCFG_TRACE_FRAME_MAX_PARAMS)
    {
        paramCount = LOSCFG_TRACE_FRAME_MAX_PARAMS;
    }

    TRACE_LOCK(intSave);
    frame->curTask   = OsTraceGetMaskTid(LOS_CurTaskIDGet());
    frame->eventType = eventType;
    TRACE_UNLOCK(intSave);
}
```

# 枚举类型命名规则

- **类型名称建议采用大驼峰命名法；枚举常量建议采用全大写的英文字母，单词之间加下划线 '_' 的方式命名:**

```
enum TraceCmd {
    TRACE_CMD_START = 1,
    TRACE_CMD_STOP,
    TRACE_CMD_SET_EVENT_MASK,
    TRACE_CMD_RECODE_DUMP,
    TRACE_CMD_MAX_CODE,
};
```

# 标示符的命名规范

1. 含义清晰，不易混淆；
2. 保持统一的命名风格。

# 注释规范

- **文件注释，可以列出：版权说明、版本号、生成日期、作者姓名、内容、功能说明、与其它文件的关系、修改日志等：**

```
/**
********************************************************************************
* @file        tem.c
* @author    YeDapeng
* @version   V1.0.0
* @date       20220209
* @brief      处理温度相关的数据
********************************************************************************
*/
```

# 注释规范

- **函数注释，描述函数功能、用法，包括输入和输出参数、函数返回值等：**

```
/**
 ********************************************************************************
 * @brief 提供华氏温度数据
 * @param
 * @return 华氏温度数值
 ********************************************************************************
 */
float GetFahTem(void)
{
        float raw = GetRawData();
        float fah = CEL_TO_FAH(raw) * 1.2f;

        return fah;
}
```

# 注释规范

- **函数注释，描述函数功能、用法，包括输入和输出参数、函数返回值等：**

```
/**
 ********************************************************************************
 * @brief 设置温度校准系数
 * @param temCof: 输入参数，温度较准系数
 * @return
 ********************************************************************************
 */

void SetTemCof(float temCof)
{
        g_temCof = temCof;
}
```

# 关于函数的编程规范

1. 函数的职责要聚焦，一个函数仅完成一件功能；
2. 避免函数过长，建议不超过50行（非空非注释行）；
3. 函数的参数个数建议不超过5个，超过了可以考虑使用结构体来组织参数，或者拆分函数；
4. 在源文件范围内声明和定义的所有函数，除非外部可见，否则应该增加static关键字，避免和其他文件或库中的相同标识符发生混淆的可能性；

# 关于函数的编程规范

5. 避免函数的代码块嵌套过深，建议不超过4层，嵌套深度指的是函数中的代码控制块（例如：if、for、while、switch等）之间互相包含的深度，｛的个数对应嵌套深度；嵌套过深会造成阅读困难。

```
void serial (void)
{
    if (!Received)
    {
        TmoCount = 0;
        switch (Buff)
        {
        case AISGFLG:
            if ((TiBuff.Count > 3)
                && ((TiBuff.Buff[0] == 0xff) || (TiBuf.Buff[0] == CurPa.ADDR)))
            {
                Flg7E = false;
                Received = true;
            }
            else
            {
                TiBuff.Count = 0;
                Flg7D = false;
                Flg7E = true;
            }
            break;
        default:
            break;
        }
    }
}
```