# 嵌入式C语言之-

# Linux中container_of宏的原理

讲师：叶大鹏

# Linux中container_of宏

```
#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE*)0)->MEMBER)

#define container_of(ptr, type, member) ({          \
        const typeof(((type *)0)->member)*__mptr = (ptr);    \
    (type *)((char *)__mptr - offsetof(type, member)); })
```

- Linux内核中，container_of宏的作用是通过结构体内某个成员的地址和该成员的名字，以及结构体类型，找到该结构体变量的首地址。

# 鸿蒙Lite os中LOS_DL_LIST_ENTRY宏

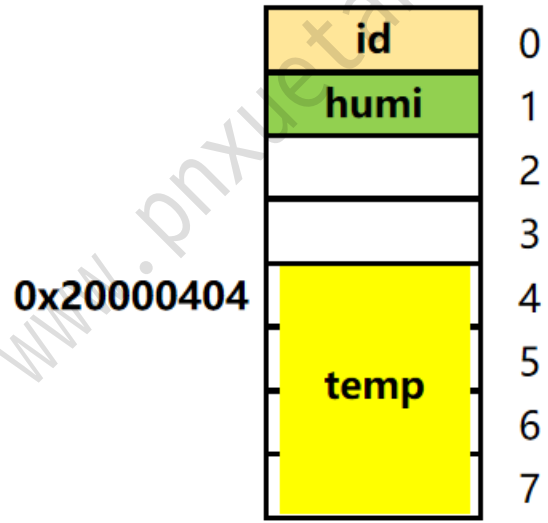
```
#define LOS_OFF_SET_OF(type, member) ((uint32_t)&((type *)0)->member)

#define LOS_DL_LIST_ENTRY(item, type, member) \
    ((type *)((uint8_t*)(item) - LOS_OFF_SET_OF(type, member)))
```

- 鸿蒙Lite os中，LOS_DL_LIST_ENTRY宏的作用是通过结构体内某个成员变量的地址和该成员的名字，以及结构体类型，找到该结构体变量的首地址。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```



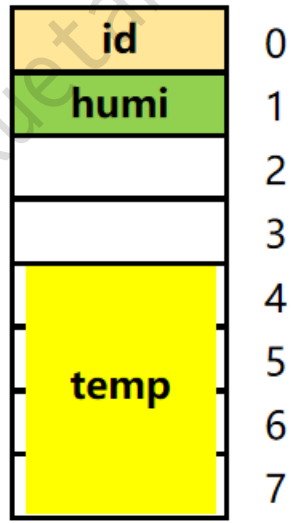

思路:

1.可以先计算成员在结构体中的偏移量:

```
(uint32_t)&((TempHumiSensor *)0)->temp
```

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```

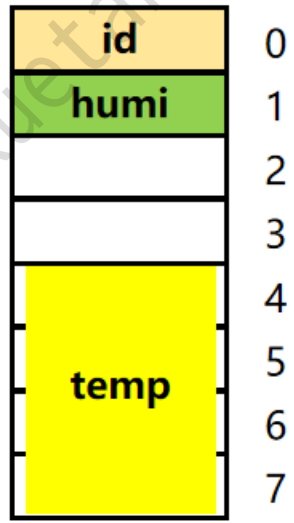



- (uint32_t)&((TempHumiSensor *)0)->temp

1）(TempHumiSensor *)0，将0转换为结构体类型指针，告诉编译器可以使用结构体类型去解释从0开始的地址空间了。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```

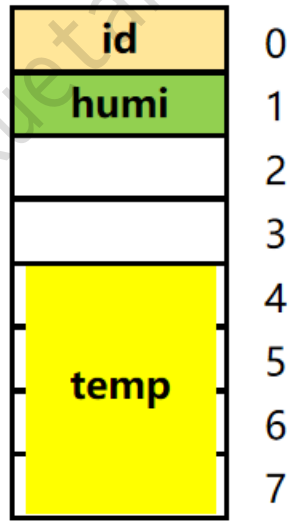



- (uint32_t)&((TempHumiSensor *)0)->temp

2）((TempHumiSensor *)0)->temp， 有了结构体类型指针，就可以去访问结构体的成员了。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```

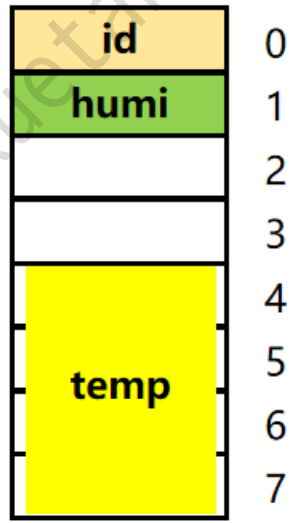



- (uint32_t)&((TempHumiSensor *)0)->temp

3) &((TempHumiSensor *)0)->temp， 能够访问结构体的成员了，可以再使用&获取这个成员的地址。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```

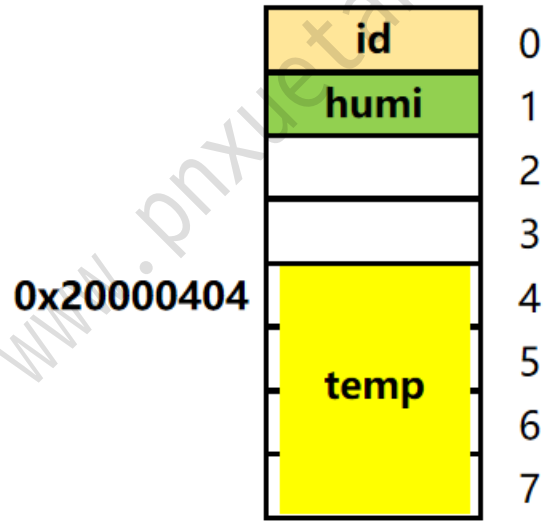



- (uint32_t)&((TempHumiSensor *)0)->temp

4) (uint32_t)&((TempHumiSensor *)0)->temp，将获得的地址转换为数值，就是偏移量数值4。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```




思路:

1.可以先计算成员在结构体中的偏移量:

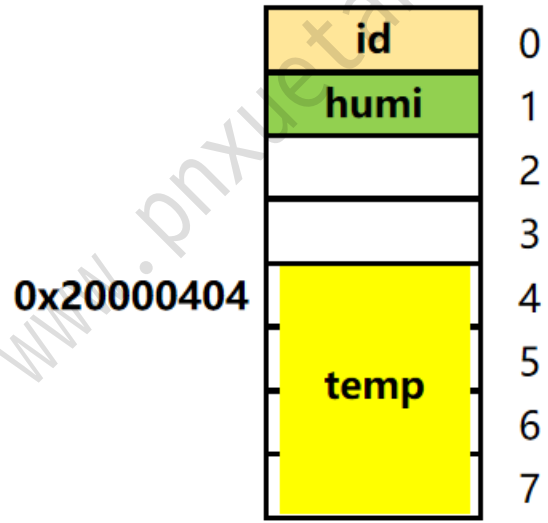
```
(uint32_t)&((TempHumiSensor *)0)->temp
```

2.根据已知的成员地址，再计算结构体变量的首地址:

```
(TempHumiSensor *)((uint8_t *)(0x20000404) - (uint32_t)&((TempHumiSensor *)0)->temp)
```

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```

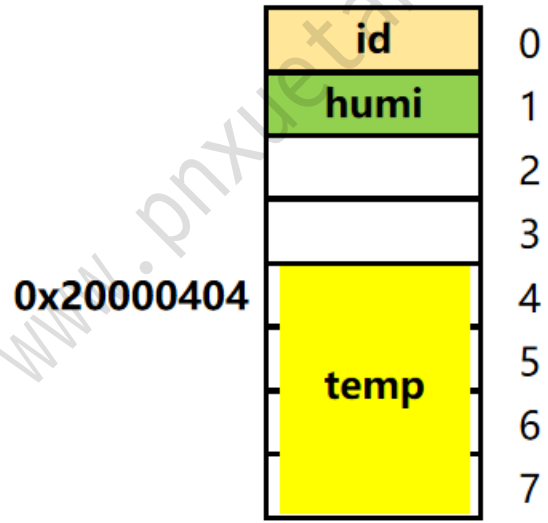



(TempHumiSensor *)((uint8_t *)(0x20000404) - (uint32_t)&((TempHumiSensor *)0)->temp)

1）(uint8_t *)(0x20000404) - (uint32_t)&((TempHumiSensor *)0)->temp，根据前面计算结果，等价于(uint8_t *)(0x20000404) - 4，因为uint8_t 的步长是1个字节，所以这里4代表的是4 * 1个字节 = 4；也就是0x20000400。

## 已知结构体内某个成员的地址，如何计算得到该结构体变量的首地址?

```
typedef struct
{
    uint8_t id;
    uint8_t humi;
    float temp;
} TempHumiSensor;
```




(TempHumiSensor *)((uint8_t *)(0x20000404) - (uint32_t)&((TempHumiSensor *)0)->temp)

2）(TempHumiSensor *)(...)，根据前面计算结果，等价于(TempHumiSensor *)(0x20000400)，这样就计算得到了首地址。

# 鸿蒙Lite os中LOS_DL_LIST_ENTRY宏的作用

```
#define LOS_OFF_SET_OF(type, member) ((uint32_t)&((type *)0)->member)

#define LOS_DL_LIST_ENTRY(item, type, member) \
    ((type *)((uint8_t *)(item)                    -   LOS_OFF_SET_OF(type, member)))
```

```
(TempHumiSensor *)((uint8_t *)(0x20000404) - (uint32_t)&((TempHumiSensor *)0)->temp)
```

1. (uint32_t)&((TempHumiSensor *)0)->temp)对应((uint32_t)&((type *)0)->member)，其中TempHumiSensor对应type，temp对应member；
2. (TempHumiSensor *)((uint8_t *)(0x20000404)对应 ((type *)((uint8_t *)(item)，其中0x20000404对应item。

# 鸿蒙Lite os中LOS_DL_LIST_ENTRY宏的作用

```
#define LOS_OFF_SET_OF(type, member) ((uint32_t)&((type *)0)->member)
#define LOS_DL_LIST_ENTRY(item, type, member) \
    ((type *)((uint8_t *)(item) - LOS_OFF_SET_OF(type, member)))
```

- 假如通过一些运算获得了结构体temp成员的地址，保存在指针变量float *pTemp里

```
TempHumiSensor *p =   LOS_DL_LIST_ENTRY(pTemp, TempHumiSensor, temp);
printf("sensor %d, humi = %d, temp = %.1f\n", p->id, p->humi, p->temp)
```

# THANK YOU!