[struct结构里的#define定义](http://blog.chinaunix.net/uid-28458801-id-3439891.html) 2012-12-14 16:25:16

分类： C/C++

**预编译（预处理）就是在正式编译之前的，为正式编译做准备，处理以#开头的指令。**

1， 在看linux内核代码时会看到某些结构体的定义中包含宏定义：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | struct i2c\_msg {      \_\_u16 addr;    /\* slave address            \*/      \_\_u16 flags;  #define I2C\_M\_TEN        0x0010    /\* this is a ten bit chip address \*/  #define I2C\_M\_RD        0x0001    /\* read data, from slave to master \*/  #define I2C\_M\_NOSTART        0x4000    /\* if I2C\_FUNC\_PROTOCOL\_MANGLING \*/  #define I2C\_M\_REV\_DIR\_ADDR    0x2000    /\* if I2C\_FUNC\_PROTOCOL\_MANGLING \*/  #define I2C\_M\_IGNORE\_NAK    0x1000    /\* if I2C\_FUNC\_PROTOCOL\_MANGLING \*/  #define I2C\_M\_NO\_RD\_ACK        0x0800    /\* if I2C\_FUNC\_PROTOCOL\_MANGLING \*/  #define I2C\_M\_RECV\_LEN        0x0400    /\* length will be first received byte \*/      \_\_u16 len;        /\* msg length                \*/      \_\_u8 \*buf;        /\* pointer to msg data            \*/  }; |

         第一次看到觉得怪怪的，虽然我知道宏定义作用于开始定义处，结束于#undef。网上也看到一些人的疑问，比如：

             a) 如果去定义这样的结构体变量，会不会导致一个宏重复定义。

             b) 既然这样没错误，那么有什么好处。

2.    通过看如下程序预编译前后的结果：

预编译前：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | struct mystr {          #define XPOS 1          #define YPOS 2          int a;          char b;  };    int main(void)  {          struct mystr ex1 = {2, 'a'};          struct mystr ex2 = {3, 'b'};          ex1.a=XPOS;          ex2.a=YPOS;            return 0;  } |

执行命令：gcc main.c -E > main.E

预编译后：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | # 1 "main.c"  # 1 ""  # 1 "<command line style="word-wrap: break-word;">"</command line>  # 1 "main.c"      struct mystr {       int a;   char b;  };    int main(void)  {   struct mystr ex1 = {2, 'a'};   struct mystr ex2 = {3, 'b'};   ex1.a=1;   ex2.a=2;         return 0;  } |

结论：

预编译时，宏被展开，宏定义处被拿掉了，所以接下来再用struct  mystr来定义变量时，里面的宏已经不存在了，所以并不会导致一个宏重复被定义。至于这样编写代码的好处，有人说是增加代码的可阅读性，让人知道这些宏只会用于这个结构体中，还有就是便于对这个结构体进行扩展。

3，对于宏编译结构体的扩张的理解如下：

#define CV\_SEQUENCE\_FIELDS() \  
**do{**

int flags; \  
int header\_size; \  
struct CvSeq\* h\_prev; \  
struct CvSeq\* h\_next; \  
struct CvSeq\* v\_prev; \  
struct CvSeq\* v\_next; \  
int total; \  
int elem\_size; \  
char\* block\_max; \  
152 CHAPTER 1. CXCORE  
char\* ptr; \  
int delta\_elems; \  
CvMemStorage\* storage; \  
CvSeqBlock\* free\_blocks; \  
CvSeqBlock\* first;

**}while(0)**

typedef struct CvSeq  
{  
     CV\_SEQUENCE\_FIELDS();  
} CvSeq;

这是opencv里面的CV\_Seq结构体的定义方式，使用宏CV\_SEQUENCE\_FIELDS()进行定义，这种定义方式有助于用户进行自行扩展。如果想在结构体里面再加入些东西，可以直接使用：  
typedef struct Myseq  
{

      CV\_SEQUENCE\_FIELDS();  
      type1 variable1;  
      type2 variable2;  
} Myseq;  
这种形式进行定义。