C 语言宏定义技巧

写好 C 语言,漂亮的宏定义很重要,使用宏定义可以防止出错,提高可移植性,可读性,方便性 等等。下面列举一些成熟软件中常用得宏定义：

1、 防止一个头文件被重复包含

```
#ifndef COMDEF_H
#define COMDEF_H
//头文件内容
#endif
```

2、 重新定义一些类型,防止由于各种平台和编译器的不同,而产生的类型字节数差异,方便移植。

```
typedef  unsigned char     boolean;     /* Boolean value type. */
typedef  unsigned long int  uint32;      /* Unsigned 32 bit value */
typedef  unsigned short     uint16;      /* Unsigned 16 bit value */
typedef  unsigned char      uint8;       /* Unsigned 8  bit value */
typedef  signed long int    int32;       /* Signed 32 bit value */
typedef  signed short       int16;       /* Signed 16 bit value */
typedef  signed char        int8;        /* Signed 8  bit value */

//下面的不建议使用
typedef  unsigned char     byte;        /* Unsigned 8  bit value type. */
typedef  unsigned short    word;        /* Unsinged 16 bit value type. */
typedef  unsigned long     dword;       /* Unsigned 32 bit value type. */
typedef  unsigned char     uint1;       /* Unsigned 8  bit value type. */
typedef  unsigned short    uint2;       /* Unsigned 16 bit value type. */
typedef  unsigned long     uint4;       /* Unsigned 32 bit value type. */
typedef  signed char       int1;        /* Signed 8  bit value type. */
typedef  signed short      int2;        /* Signed 16 bit value type. */
typedef  long int          int4;        /* Signed 32 bit value type. */
typedef  signed long       sint31;      /* Signed 32 bit value */
typedef  signed short      sint15;      /* Signed 16 bit value */
typedef  signed char       sint7;       /* Signed 8  bit value */
```

3、 得到指定地址上的一个字节或字

```
#define  MEM_B( x )  ( *( (byte *) (x) ) )
#define  MEM_W( x )  ( *( (word *) (x) ) )
```

4、 求最大值和最小值

```
#define  MAX( x, y ) ( ((x) > (y)) ? (x) : (y) )
#define  MIN( x, y ) ( ((x) < (y)) ? (x) : (y) )
```

5、 得到一个 field 在结构体(struct)中的偏移量

```
#define FPOS( type, field ) /
/*lint -e545 */ ( (dword) &(( type *) 0)-> field ) /*lint +e545 */
```

6、 得到一个结构体中 field 所占用的字节数

```
#define FSIZ( type, field ) sizeof( ((type *) 0)->field )
```

7、 按照 LSB 格式把两个字节转化为一个 Word

```
#define  FLIPW( ray ) ( (((word) (ray)[0]) * 256) + (ray)[1] )
```

8、 按照 LSB 格式把一个 Word 转化为两个字节

```
#define  FLOPW( ray, val )/
            (ray)[0] = ((val) / 256); /
            (ray)[1] = ((val) & 0xFF)
```

9、 得到一个变量的地址（word 宽度）

```
#define  B_PTR( var )  ( (byte *) (void *) &(var) )
#define  W_PTR( var )  ( (word *) (void *) &(var) )
```

10、 得到一个字的高位和低位字节

```
#define  WORD_LO(xxx)  ((byte) ((word)(xxx) & 255))
#define  WORD_HI(xxx)  ((byte) ((word)(xxx) >> 8))
```

11、返回一个比 X 大的最接近的 8 的倍数

```
#define RND8( x )       ((((x) + 7) / 8 ) * 8 )
```

12、将一个字母转换为大写

```
#define  UPCASE( c ) ( ((c) >= ''a'' && (c) <= ''z'') ? ((c) - 0x20) : (c) )
```

13、判断字符是不是 10 进值的数字

```
#define  DECCHK( c ) ((c) >= ''0'' && (c) <= ''9'')
```

14、判断字符是不是 16 进值的数字

```
#define  HEXCHK( c ) ( ((c) >= ''0'' && (c) <= ''9'') ||/
                       ((c) >= ''A'' && (c) <= ''F'') ||/
                       ((c) >= ''a'' && (c) <= ''f'') )
```

15、防止溢出的一个方法

```
#define  INC_SAT( val )  (val = ((val)+1 > (val)) ? (val)+1 : (val))
```

16、返回数组元素的个数

```
#define  ARR_SIZE( a )  ( sizeof( (a) ) / sizeof( (a[0]) ) )
```

17、返回一个无符号数 n 尾的值 MOD_BY_POWER_OF_TWO(X,n)=X%(2^n)

```
#define MOD_BY_POWER_OF_TWO( val, mod_by ) /
                                ( (dword)(val) & (dword)((mod_by)-1) )
```

18、对于 IO 空间映射在存储空间的结构,输入输出处理

```
#define inp(port)          (*((volatile byte *) (port)))
#define inpw(port)         (*((volatile word *) (port)))
#define inpdw(port)        (*((volatile dword *)(port)))
#define outp(port, val)    (*((volatile byte *) (port)) = ((byte) (val)))
#define outpw(port, val)   (*((volatile word *) (port)) = ((word) (val)))
#define outpdw(port, val)  (*((volatile dword *) (port)) = ((dword) (val)))
```

19、防止宏定义使用错误

用小括号包含，例如：#define ADD(a,b) （a+b）

用 do{}while(0)语句包含多语句防止错误，例如：

```
#difne DO(a,b) a+b;/ a++;

应用时：if(….)
          DO(a,b); //产生错误
        else
```

解决方法: #difne DO(a,b) do{a+b;/ a++;}while(0)