1. 引子   
int a;     
int size = 8;  
      
如果让a为(size =8)的整数倍表示成二进制应是什么样子呢？那就是让这个数表示成二进制时的最后三位为0.  
而要达到这一目标，只要下面这个数与它进行与运算就可以了:  
  11111111 11111111 11111111 11111000     
而上面这个数实际下就是 ~ (size - 1)，可以将该数称为size的对齐掩码size\_mask.     
      
可这样做求出的是比a小的那个最大的8的倍数. 如果要求出比a大的是不是需要加上8就可以了?   
可是如果a本身就是8的倍数, 这样加8不就错了吗, 所以在a基础上加上 size - 1, 然后与size的对齐掩码进行与运算.   
  
这样, 我们可以定义下面的宏, 用于计算一个数a以size为倍数的前后两个值：  
#define alignment\_down(a, size) (a & (~(size-1)) )  
#define alignment\_up(a, size) ((a+size-1) & (~ (size-1)))   
    
例如:   
a=0, size=8,  则alignment\_down(a,size)=0, alignment\_up(a,size)=0.  
a=6, size=8,  则alignment\_down(a,size)=0, alignment\_up(a,size)=8.  
a=8, size=8,  则alignment\_down(a,size)=8, alignment\_up(a,size)=8.  
a=14, size=8, 则alignment\_down(a,size)=8, alignment\_up(a,size)=16.   
    
注意：size应当为2的n次方, 即2, 4, 8, 16, 32, 64, 128, 256, 1024, 2048, 4096.....   
    
2. 在linux中的应用  
上面的计算方法在linux等代码中也常常可以看到,下面给出几个例子.   
    
例１：当分配地址addr时, 要将该地址以size为倍数对齐, 而且要得到是比addr大的值, 则使用\_ALIGN宏：  
/\* align addr on a size boundary - adjust address up if needed -- Cort \*/  
#define \_ALIGN(addr,size)   (((addr)+(size)-1)&(~((size)-1)))   
  
例2：与页面对齐相关的宏  
#define PAGE\_SIZE               4096  
#define PAGE\_MASK               (~(PAGE\_SIZE-1))   
/\* to align the pointer to the (next) page boundary \*/  
#define PAGE\_ALIGN(addr)        (((addr)+PAGE\_SIZE-1) & PAGE\_MASK)   
    
例3：与skb分配时对齐相关的宏  
#define SKB\_DATA\_ALIGN(X)   (((X) + (SMP\_CACHE\_BYTES - 1)) & ~(SMP\_CACHE\_BYTES - 1))   
    
struct sk\_buff \*\_\_alloc\_skb(unsigned int size, gfp\_t gfp\_mask, int fclone)  
{  
    ......  
   
    /\* Get the HEAD \*/  
    skb = kmem\_cache\_alloc(cache, gfp\_mask & ~\_\_GFP\_DMA);  
    if (!skb)  
         goto out;   
    
    /\* Get the DATA. Size must match skb\_add\_mtu(). \*/  
    size = SKB\_DATA\_ALIGN(size);  
    data = kmalloc\_track\_caller(size + sizeof(struct skb\_shared\_info),  gfp\_mask);  
    if (!data)  
         goto nodata;   
    ......  
}