<https://blog.csdn.net/21aspnet/article/details/160037>

位移位运算符是将数据看成二进制数，对其进行向左或向右移动若干位的运算。位移位运算符分为左移和右移两种，均为双目运算符。第一运算对象是移位对象，第二个运算对象是所移的二进制位数。  
　　位移位运算符的运算对象、运算规则与结果、结合性如表2-16所示。  
  
　　移位时，移出的位数全部丢弃，移出的空位补入的数与左移还是右移花接木有关。如果是左移，则规定补入的数全部是0；如果是右移，还与被移位的数据是否带符号有关。若是不带符号数，则补入的数全部为0；若是带符号数，则补入的数全部等于原数的最左端位上的原数(即原符号位)。具体移位规则如下所示。  
  
　　位移位运算符的优先级如下：  
　　　·算术运算符　优先于　位移位运算符 优先于 关系运算符  
　　　·位移位运算符是同级别的，结合性是自左向右  
　　例如，设无符号短整型变量a为0111(对应二进制数为0000000001001001),  
　　　则：a<<3 结果为01110(对应二进制数为0000001001001000)，a不变  
　　　　　a>>4 结果为04　(对应二进制数为0000000000000100)，a不变  
　　又如，设短整型变量a为-4(对应二进制数为 1111111111111100),  
　　　则：a<<3 结果为-32(对应二进制数为1111111111100000)，a不变  
　　　　　a>>4 结果为-1(对应二进制数为1111111111111111)，a不变

C语言里的左移和右移运算

2006-09-30 13:52

先说左移,左移就是把一个数的所有位都向左移动若干位,在C中用<<运算符.例如:

int i = 1;  
i = i << 2;    //把i里的值左移2位

也就是说,1的2进制是000...0001(这里1前面0的个数和int的位数有关,32位机器,gcc里有31个0),左移2位之后变成 000... 0100,也就是10进制的4,所以说左移1位相当于乘以2,那么左移n位就是乘以2的n次方了(有符号数不完全适用,因为左移有可能导致符号变化,下面解释原因)

需要注意的一个问题是int类型最左端的符号位和移位移出去的情况.我们知道,int是有符号的整形数,最左端的1位是符号位,即0正1负,那么移位的时候就会出现溢出,例如:

int i = 0x40000000; //16进制的40000000,为2进制的01000000...0000  
i = i << 1;

那么,i在左移1位之后就会变成0x80000000,也就是2进制的100000...0000,符号位被置1,其他位全是0,变成了int类型所能表示的最小值,32位的int这个值是-2147483648,溢出.如果再接着把i左移1位会出现什么情况呢?在C语言中采用了丢弃最高位的处理方法,丢弃了1之后,i的值变成了0.

左移里一个比较特殊的情况是当左移的位数超过该数值类型的最大位数时,编译器会用左移的位数去模类型的最大位数,然后按余数进行移位,如:

int i = 1, j = 0x80000000; //设int为32位  
i = i << 33;     // 33 % 32 = 1 左移1位,i变成2  
j = j << 33;     // 33 % 32 = 1 左移1位,j变成0,最高位被丢弃

在用gcc编译这段程序的时候编译器会给出一个warning,说左移位数>=类型长度.那么实际上i,j移动的就是1位,也就是33%32 后的余数.在gcc下是这个规则,别的编译器是不是都一样现在还不清楚.

总之左移 就是: 丢弃最高位,0补最低位

再说右移,明白了左移的道理,那么右移就比较好理解了.

右移的概念和左移相反,就是往右边挪动若干位,运算符是>>.

右移对符号位的处理和左移不同,对于有符号整数来说,比如int类型,右移会保持符号位不变,例如:

int i = 0x80000000;  
i = i >> 1;    //i的值不会变成0x40000000,而会变成0xc0000000

就是说,符号位向右移动后,正数的话补0,负数补1,也就是汇编语言中的算术右移.同样当移动的位数超过类型的长度时,会取余数,然后移动余数个位.

       负数10100110 >>5(假设字长为8位)，则得到的是    11111101

总之,在C中,左移是逻辑/算术左移(两者完全相同),右移是算术右移,会保持符号位不变 .实际应用中可以根据情况用左/右移做快速的乘 /除运算,这样会比循环效率高很多.

  在很多系统程序中常要求在位(bit)一级进行运算或处理。Ｃ语言提供了位运算的功能， 这使得Ｃ语言也能像汇编语言一样用来编写系统程序。  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━   
操作符 作用   
────────────────────────────   
& 位逻辑与   
| 位逻辑或   
^ 位逻辑异或   
- 位逻辑反   
>> 右移   
<< 左移   
━━━━━━━━━━━━━━━━━━━━━━━━━━━━   
         按位运算是对字节或字中的实际位进行检测、设置或移位, 它只适用于字符型和整数型变量以及它们的变体, 对其它数据类型不适用。  
         我们要注意区分位运算和逻辑运算。

         1. 按位与运算 按位与运算符"&"是双目运算符。其功能是参与运算的两数各对应的二进位相与。只有对应的两个二进位均为1时，结果位才为1 ，否则为0。参与运算的数以补码方式出现。  
例如：9&5可写算式如下： 00001001 (9的二进制补码)&00000101 (5的二进制补码)　00000001 (1的二进制补码)可见9&5=1。  
         按位与运算通常用来对某些位清0或保留某些位。例如把a 的高八位清 0 ， 保留低八位， 可作 a&255 运算 ( 255 的二进制数为0000000011111111)。  
main(){  
int a=9,b=5,c;  
c=a&b;  
printf("a=%d\nb=%d\nc=%d\n",a,b,c);  
}

2. 按位或运算 按位或运算符“|”是双目运算符。其功能是参与运算的两数各对应的二进位相或。只要对应的二个二进位有一个为1时，结果位就为1。参与运算的两个数均以补码出现。  
例如：9|5可写算式如下： 00001001|00000101  
00001101 (十进制为13)可见9|5=13  
main(){  
int a=9,b=5,c;  
c=a|b;  
printf("a=%d\nb=%d\nc=%d\n",a,b,c);  
}

3. 按位异或运算 按位异或运算符“^”是双目运算符。其功能是参与运算的两数各对应的二进位相异或，当两对应的二进位相异时，结果为1。参与运算数仍以补码出现，例如 9^5可写成算式如下： 00001001^00000101 00001100 (十进制为12)  
main(){  
int a=9;  
a=a^15;  
printf("a=%d\n",a);  
}

4. 求反运算 求反运算符～为单目运算符，具有右结合性。 其功能是对参与运算的数的各二进位按位求反。例如～9的运算为： ~(0000000000001001)结果为：1111111111110110

5. 左移运算 左移运算符“<<”是双目运算符。其功能把“<< ”左边的运算数的各二进位全部左移若干位，由“<<”右边的数指定移动的位数，高位丢弃，低位补0。例如： a<<4 指把a的各二进位向左移动4位。如a=00000011(十进制3)，左移4位后为00110000(十进制48)。

6. 右移运算 右移运算符“>>”是双目运算符。其功能是把“>> ”左边的运算数的各二进位全部右移若干位，“>>”右边的数指定移动的位数。例如：设 a=15，a>>2　表示把000001111右移为00000011(十进制3)。应该说明的是，对于有符号数，在右移时，符号位将随同移动。当为正数时， 最高位补0，而为负数时，符号位为1，最高位是补0或是补1 取决于编译系统的规定。

main(){  
unsigned a,b;  
printf("input a number: ");  
scanf("%d",&a);  
b=a>>5;  
b=b&15;  
printf("a=%d\tb=%d\n",a,b);  
}  
请再看一例!  
main(){  
char a='a',b='b';  
int p,c,d;  
p=a;  
p=(p<<8)|b;  
d=p&0xff;  
c=(p&0xff00)>>8;  
printf("a=%d\nb=%d\nc=%d\nd=%d\n",a,b,c,d);  
}

当进行按位与或时，最好使用16进制，在程序中这样表示：0x01 表示0000 0001  
所以，字符类型a的最高位强制1可以这样：a=a|0x80。其他的可以依次类推！

阅读全文

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/21aspnet/article/details/160037