C语言编程常见问题解答之位(bit)和字节(byte) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/285/2021_2022_C_E8_AF_AD_E8_A8_80_E7_BC_96_c97_285302.htm 位指的是二进制系统中的一位，它是最小的信息单位。位的用处可以从两方面去分析：第一，计算机对位的值可以有任意多种解释，例如表示"yes ’ ’ 或"no ”，或者表示磁盘是否已插入驱动器，或者表示某个鼠标键是否被按下；第二，将若干位的值连接起来后，就可以表示更复杂的数据，而且每增加一位，可以表示的可能的值的数目就会增加一倍。 换句话说，一位可以表示两种可能的值，即“ O ”和“ 1 ”；两位可以表示2×2或4种可能的值，即“ 00 ”，“ 01 ”，“ 10 ”和“ 11 ”；类似地，三位可以表示2×2×2或8种可能的值……。对计算机来说，位的这种特性既是最有力的支持因为很复杂的数据(例如本书内容)可以被分解为位的表示后存储起来，又是最大的限制因为在现实生活中许多事物的值是不精确的，这样的值无法用数目有限的若干位来表示。 程序员始终必须清楚每一项数据需要用多少位来表示。因为位作为单位太小，所以为了方便起见，大多数计算机所处理的信息单位是被称为字节的位块。字节是大多数计算机中最小的可寻址的信息单位，这意味着计算机给每一个字节的信息都赋予一个地址，并且一次只能存取一个字节的信息。一个字节中的位的数目可以是任意的，并且在不同的计算机中可以不同。最常见的情况是每个字节中有8位，即可以存放256个不同的值。8位这样的长度非常适合于存放表示ASCII(the American Standard Code for Information Interchange)字符的数据。 下述程序可以显示空格

符以后的ASCII字符和PC机的图形字符集： # include void main (void). void main() { /" Display ASCII char set " / unsigned char space = ’ ’. /* Start with SPACE char = 8 bits only * / int ctr = 0. printf(" ASCII Characters\n")amp.1；为了置位所需的位，可以让数据和屏蔽字进行按位或操作(C的按位或运算符为|)。例如，你可以这样置位flags的最低位： flags = flags | 1；或者这样： flags |= 1；为了清除所需的位，可以让数据和对屏蔽字按位取反所得的值进行按位与操作。例如，你可以这样清除flags的最低位： flags = flagsamp.=~1 . 有时，用宏来处理标志会更方便，例10．2中的程序就是通过一些宏简化了位操作。例10．2 能使标志处理更方便的宏 /* Bit Masking * / / * Bit masking can be used to switch a character between lowercase and uppercase * / #define BIT_POS(N) (1U amp.= - (F)) #define TST_FLAGCN,F) ((N) laquo.(N)) #define BIT_SHIFTR(B,N) ((unsigned)(B)amp.= ~(F)) #define GET_MFLAG(N,F) ((N) &amp. (F)) # include void main() { unsigned char ascii_char = ’A ’. /* char = 8 bits only */ int test_nbr = 10. printf("Starting character = %c\n" , ascii_char). /" The 5th bit position determines if the character is uppercase or lowercase. 5th bit = 0 - Uppercase 5th bit = 1- Lowercase * / printf ("\nTurn 5th bit on = %c\n" , SET_FLAG(ascii_char, BIT_POS(5))). printf ("Turn 5th bit off = %c\n\n",CLR_FLAG(ascii_char, BIT_POS(5))). printf ("Look at shifting bits\n"). printf (" = = = = = = = = = = = = = = = =\n"). printf ("Current value = %d\n" , test_nbr)i printf ("Shifting one position left = %d\n" , test_nbr = BIT_SHIFTL(test_nbr, 1)). printf ("Shifting two positions right = %d\n" , BIT_SHIFTR(test_nbr, 2)).