<https://blog.csdn.net/qq_27522735/article/details/53374765>

1. memset()函数原型是extern void \*memset(void \*buffer, int c, int count)        buffer：为指针或是数组,

              c：是赋给buffer的值,

       count：是buffer的长度.

       这个函数在socket中多用于清空数组.如:原型是memset(buffer, 0, sizeof(buffer))

       Memset用来对一段内存空间全部设置为某个字符，一般用在对定义的字符串进行初始化为‘ ’或‘/0’；

      例:char a[100];memset(a, '/0', sizeof(a));

    memset可以方便的清空一个结构类型的变量或数组。

如：

struct sample\_struct  
{  
char csName[16];  
int iSeq;  
int iType;  
};

对于变量：  
struct sample\_strcut stTest;

一般情况下，清空stTest的方法：

stTest.csName[0]='/0';  
stTest.iSeq=0;  
stTest.iType=0;

用memset就非常方便：  
memset(&stTest,0,sizeof(struct sample\_struct));

如果是数组：

struct sample\_struct TEST[10];  
则  
memset(TEST,0,sizeof(struct sample\_struct)\*10);

2.提问：“将s所指向的某一块内存中的每个字节的内容全部设置为ch指定的ASCII值,块的大小由第三个参数指定,这个函数通常为新申请的内存做初始化工作。用 法: void \*memset(void \*s, char ch, unsigned n);”

//“将s所指向的某一块内存中的每个字节的内容全部设置为ch指定的ASCII值”干什么？为什么说“这个函数通常为新申请的内存做初始化工作。”？

答：刚分配的内存空间，或者是被你用过的内存空间里面的数据是不固定的为了避免这些无用的数据给自己的程序带来影响，可以用memset把这些内存里面的数据置成某个数值一般情况下是置0,当然，如果你的程序不会受这些无用数据影响，就不用做这个工作了所谓“初始化”，当然是指将你定义的变量或申请的空间赋予你所期望的值，例如语句int i=0;就表明定义了一个变量i,并初始化为0；如果int j=5;就表明定义了一个变量j，并初始化为5。  
  
       但是对于大块儿内存的分配，这种方法当然不行，例如int arr[100];定义了数组arr,包含100个元素，如果你写成int arr[100]=0;想将数组全部内容初始化为0，是不行的，连编译都不能通过。这种情况的初始化，有两种方法，一种是一个一个的初始化，如for(int i=0;i<100;i++)arr[i]=0;就完成了数组的初始化。另一种方法，就是使用memset：一个语句就够了--memset(arr,0,sizeof(int)\*100);  
各参数解释如下：arr是数组的首地址，0就是要讲这些地址的内容赋值为0，sizeof(int)求出int类型的长度，乘以100就表示arr数组的整个长度。

         当然，如果用malloc分配的内存，一般只能使用memset来初始化了，用第一种初始化方法明显不合适。

         例：char ch[10]  
         比如memset(ch,0,8),就是把数组ch前八项置为零，后面的不一定为零。比如刚开始ch[1]='z',ch[8]='a',ch[9]='b',经过memset后，ch[1]为零了，而ch[8],ch[9]都不变.

3.memset函数详细说明   
       1)void \*memset(void \*s,int c,size\_t n)  
        总的作用：将已开辟内存空间 s 的首 n 个字节的值设为值 c。

       2).memset() 函数常用于内存空间初始化。如：  
           char str[100];  
           memset(str,0,100);

       3).memset可以方便的清空一个结构类型的变量或数组。

           如：  
           struct sample\_struct{  
                      char csName[16];  
                       int iSeq;  
                       int iType;  
           };

           对于变量：  
           struct sample\_strcut stTest;

           一般情况下，清空stTest的方法：  
           stTest.csName[0]='/0';  
           stTest.iSeq=0;  
           stTest.iType=0;

           用memset就非常方便：  
           memset(&stTest,0,sizeof(struct sample\_struct));

           如果是数组：  
           struct sample\_struct TEST[10];  
           则  
           memset(TEST,0,sizeof(struct sample\_struct)\*10);

           #include <mem.h>  
           void\* memset(void\* s, int c, size\_t n){  
                     unsigned char\* p = (unsigned char\*) s;

                     while (n > 0) {  
                                \*p++ = (unsigned char) c;  
                                  --n;  
                      }

                     return s;  
           }

        memset()的函数， 它可以一字节一字节地把整个数组设置为一个指定的值。 memset()函数在mem.h头文件中声明，它把数组的起始地址作为其第一个参数，第二个参数是设置数组每个字节的值，第三个参数是数组的长度(字节数，不是元素个数)。其函数原型为：  
　　　　void \*memset(void\*，int，unsigned)；  
　　其中void\*表示地址。  
　　例如，下面的代码用数组做参数传递给标准函数memset()，以让其将数组设置成全0：  
　　　　#include<mem.h>  
　　　　void main()  
　　　　{  
　　　　　int ia1[50]；  
　　　　　int ia2[500]；  
　　　　　memset(iai,0,50\*sizeof(int))；  
　　　　　memset(ia2,0,500\*sizeof(int))；  
　　　　　//  
　　　　}  
　　memset()的第一个实参是数组名，数组名作参数即数组作参数，它仅仅只是一个数组的起始地址而已。  
　　在函数memset()栈区，从返回地址往上依次为第1，2，3个参数。第1个参数中的内容是main()函数中定义的数组ia1的起始地址。第2个参数是给数组设置的值(0)，第3个参数是数组的长度(50\*2)。函数返回时，main()函数的数组中内容全置为0。  
栈  
0 && image.height>0){if(image.width>=700){this.width=700;this.height=image.height\*700/image.width;}}">

memset函数详细说明

1。void \*memset(void \*s,int c,size\_t n)  
总的作用：将已开辟内存空间 s 的首 n 个字节的值设为值 c。

2。例子  
＃include

void main(){  
char \*s="Golden Global View";

clrscr();

memset(s,'G',6);  
printf("%s",s);

getchar();  
return 0;  
}　  
3。memset() 函数常用于内存空间初始化。如：  
char str[100];  
memset(str,0,100);

4。memset()的深刻内涵：用来对一段内存空间全部设置为某个字符，一般用在对定义的字符串进行初始化为‘ ’或‘/0’；例:char a[100];memset(a, '/0', sizeof(a));

memcpy用来做内存拷贝，你可以拿它拷贝任何数据类型的对象，可以指定拷贝的数据长度；例：char a[100],b[50]; memcpy(b, a, sizeof(b));注意如用sizeof(a)，会造成b的内存地址溢出。

strcpy就只能拷贝字符串了，它遇到'/0'就结束拷贝；例：char a[100],b[50];strcpy(a,b);如用strcpy(b,a)，要注意a中的字符串长度（第一个‘/0’之前）是否超过50位，如超过，则会造成b的内存地址溢出。

5.补充：  
memset可以方便的清空一个结构类型的变量或数组。

如：  
struct sample\_struct  
{  
char csName[16];  
int iSeq;  
int iType;  
};

对于变量  
struct sample\_strcut stTest;

一般情况下，清空stTest的方法：  
stTest.csName[0]='/0';  
stTest.iSeq=0;  
stTest.iType=0;

用memset就非常方便：  
memset(&stTest,0,sizeof(struct sample\_struct));

如果是数组：  
struct sample\_struct TEST[10];  
则  
memset(TEST,0,sizeof(struct sample\_struct)\*10);

6。strcpy  
原型：extern char \*strcpy(char \*dest,char \*src);  
用法：＃i nclude  
功能：把src所指由NULL结束的字符串复制到dest所指的数组中。  
说明：src和dest所指内存区域不可以重叠且dest必须有足够的空间来容纳src的字符串。  
返回指向dest的指针。  
memcpy  
原型：extern void \*memcpy(void \*dest, void \*src, unsigned int count);  
用法：＃i nclude  
功能：由src所指内存区域复制count个字节到dest所指内存区域。  
说明：src和dest所指内存区域不能重叠，函数返回指向dest的指针。  
memset  
原型：extern void \*memset(void \*buffer, int c, int count);  
用法：＃i nclude  
功能：把buffer所指内存区域的前count个字节设置成字符c。  
说明：返回指向buffer的指针。

**memcpy**用来做内存拷贝，你可以拿它拷贝任何数据类型的对象，可以指定拷贝的数据长度。

例：char a[100],b[50]; memcpy(b, a, sizeof(b));注意如用sizeof(a)，会造成b的内存地址溢出。

**Strcpy**就只能拷贝字符串了，它遇到'/0'就结束拷贝。

例：char a[100],b[50];strcpy(a,b);如用strcpy(b,a)，要注意a中的字符串长度（第一个‘/0’之前）是否超过50位，如超过，则会造成b的内存地址溢出。

str也可以用用个参数的strncpy(a,b,n)

========================================================

memset主要应用是初始化某个内存空间。  
memcpy是用于copy源空间的数据到目的空间中。  
strcpy用于字符串copy,遇到‘/0’，将结束。

如果你理解了这些，你应该知道他们的区别：例如你初始化某块空间的时候，用到memcpy，那么应该怎么写，是不是显得很笨。  
int m[100]  
memset((void\*)m,0x00,sizeof(int)\*100);//Ok！  
memcpy((void\*)m,"/0/0/0/0....",sizeof(int)\*100);//it’s wrong.

memset用法详解(转）

|  |
| --- |
| **memest原型 (please type "man memset" in your shell)**  **void \*memset(void \*s,   int c, size\_t n);**  memset:作用是在一段内存块中填充某个给定的值，它对较大的结构体或数组进行清零操作的一种最快方法。  **常见的三种错误**  **第一: 搞反了c 和 n的位置.**  一定要记住 如果要把一个char a[20]清零, 一定是 memset(a, 0, 20)  而不是 memset(a, 20,   0)  **第二: 过度使用memset, 我想这些程序员可能有某种心理阴影, 他们惧怕未经初始化的内存, 所以他们会写出这样的代码:**  char buffer[20];  memset(buffer, 0, sizeof((char)\*20));  strcpy(buffer, "123");  这里的memset是多余的. 因为这块内存马上就被覆盖了, 清零没有意义.  **第三: 其实这个错误严格来讲不能算用错memset, 但是它经常在使用memset的场合出现**  int some\_func(struct something \*a){  …  …  memset(a, 0, sizeof(a));  …  }  **问**:**为何要用memset置零?**memset( &Address, 0, sizeof(Address))；经常看到这样的用法，其实不用的话，分配数据的时候，剩余的空间也会置零的。  **答:1.**如果不清空，可能会在[测试](http://lib.csdn.net/base/softwaretest)当中出现野值。 你做下面的试验看看结果()  char buf[5]；  CString str,str1； //memset(buf,0,sizeof(buf))； for(int i = 0；i<5；i++) { str.Format(“%d “,buf[i])； str1 +=str ； } TRACE(“%s/r/n“,str1)  **2**.其实不然！特别是对于字符指针类型的，剩余的部分通常是不会为0的，不妨作一个试验，定义一个字符数组，并输入一串字符，如果不用memset实现清零，使用MessageBox显示出来就会有乱码（0表示NULL，如果有，就默认字符结束，不会输出后面的乱码）  **问:**  **如下demo是可以的，能把数组中的元素值都设置成字符1**， #include <iostream> #include <cstring> using namespace std; int main() {      char a[5];      memset(a,'1',5);      for(int i = 0;i < 5;i++)        cout<<a[i]<<"   ";      system("pause");      return 0; } **而，如下程序想吧数组中的元素值设置成1，却是不可行的** #include <iostream> #include <cstring> using namespace std; int main() {      int a[5];      memset(a,1,5);//这里改成memset(a,1,5 \*sizeof(int))也是不可以的      for(int i = 0;i < 5;i++)        cout<<a[i]<<"   ";      system("pause");      return 0; } **问题是：**  **1，第一个程序为什么可以，而第二个不行， 2，不想要用for，或是while循环来初始化int a[5];能做到吗？（有没有一个像memset（）这样的函数初始化）**  **答:**  1.因为第一个程序的数组a是字符型的，字符型占据内存大小是1Byte，而memset函数也是以字节为单位进行赋值的，所以你输出没有问题。而第二个程序a是整型的，使用memset还是按字节赋值，这样赋值完以后，每个数组元素的值实际上是0x01010101即十进制的16843009。你看看你输出结果是否这样？  2.如果用memset(a,1,20); 就是对a指向的内存的20个字节进行赋值，每个都用ASCII为1的字符去填充，转为二进制后，1就是00000001,占一个字节。一个INT元素是4字节，合一起就是1000000010000000100000001，就等于16843009，就完成了对一个INT元素的赋值了。 |