memset

memset是计算机中C/C++语言函数。将s所指向的某一块内存中的前[illegible]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[illegible]作， 其返回值为指向s的指针。

中文名	memset函数	应用学科	计
别　称	char型初始化函数	适用领域范围	C
表达式	memset(a,0,sizeof(a))	头文件	<memory.h>或<string.h>

目录

1 函数介绍

void *memset(void *s, int ch, size_t n);

函数解释：将s中前n个字节 （typedef unsigned int size_t ）用 ch 替换并返回 s 。

memset：作用是在一段内存块中填充某个给定的值，它是对较大的结构体或数组进行清零操作的一种最快方法 [1]。

2 常见错误

第一： 搞反了 ch 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[4];
memset(buffer,0,sizeof(char)*4);
strcpy(buffer,"123");
//////////////////////////////////////////////////////////////
//////////"123"中最后隐藏的'\0'占一位，总长4位。
```

这里的memset是多余的. 因为这块内存马上就被全部覆盖，清零没有意义.

另：以下情况并不多余，因某些编译器分配空间时，内存中默认值并不为0：

```
char buffer[20];
memset(buffer,0,sizeof(char)*20);
memcpy(buffer,"123",3);
//////////////////////////////////////////////////////////////
//////////这一条的memset并不多余，memcpy并没把buffer全部覆盖，如果没有memset，
//////////用printf打印buffer会有乱码甚至会出现段错误。
//////////如果此处是strcpy(buffer,"123");便不用memset,strcpy虽然不会覆盖buffer但是会
```

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图册

相关函数 纠错


char


拷贝构造函数


scanf

sprintf


strcpy

strcat


strcmp


atoi


fread

词条统计

浏览次数：1005076次
编辑次数：107次 历史版本
最近更新：2015-02-11
创建者：kakatakaka

百科消息：

安全用耳，保护听力的正确方式
3D恐龙博物馆，邀你一起来体验
【公告】词条打标签功能上线啦！
景宁畲族自治县畲族博物馆上线啦！

```
…
…
memset(a,0,sizeof(a));
…
}
```

这里错误的原因是VC函数传参过程中的指针降级，导致sizeof(a)，返回的是一个 something*指针类型大小的的字节数，如果是32位，就是4字节。

3 常见问题

编辑

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，可能会在测试当中出现野值。你做下面的试验看看结果()

```
#include <iostream>
#include "string.h"
#include <afx.h>
using namespace std;
int main()
{
    char buf[5];
    CString str;
    CString str1;
    CString str2;
    memset(buf,0,sizeof(buf));
    for(int i=0;i<5;i++)
    {
        str.Format("%d",buf[i]);
        str1+=str;
    }
    str2.Format("%d",str1);
    cout<<str2<<endl;
    system("pause");
    return 0;
}
```

这样写，有没有memset，输出都是一样

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，如下程序想把数组中的元素值设置成1，却是不可行的

```
#include <iostream>
#include <cstring>
#include <windows.h>
using namespace std;
int main()
{
    int a[5];
    memset(a,1,20）;//如果这里改成memset(a,1,5*sizeof(int))也可以，因为memset按字节
    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？

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。

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）

如果用memset(a,1,20);（实际上与memset(a,1,5*sizeof(int))结果是一样的）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，合一起是0000 0001,0000 0001,0000 0001,0000 0001，转化成十六进制就是0x01010101，就等于16843009，就完成了对一个INT元素的赋值了。

4 程序范例

编辑

```
#include <string.h>
#include <stdio.h>
#include <memory.h>

int main(void)
{
    char buffer[]="Helloworld\n";
    printf("Buffer before memset:%s\n",buffer);
    memset(buffer,'*',strlen(buffer));
    printf("Buffer after memset:%s\n",buffer);
    return 0;
}
```

输出结果：

```
Buffer before memset:Helloworld
Buffer after memset:**********
```

编译平台：

Microsoft Visual C++6.0

也不一定就是把内容全部设置为ch指定的ASCⅡ值，而且该处的ch可为int或者其他类型，并不一定要是char类型。例如下面这样：

```
int array[5]={1,4,3,5,2};
for(int i=0;i<5;i++)
cout<<array[i]<<"";
cout<<endl;

memset(array,0,5*sizeof(int));
for(int k=0;k<5;k++)
cout<<array[k]<<"";
cout<<endl;
```

输出的结果就是：

```
14352
00000
```

后面的表大小的参数是以字节为单位，所以，对于int或其他的就并不是都乘默认的1（字符型）了。而且不同的机器上int的大小也可能不同，所以最好用sizeof()。

要注意的是，memset是对字节进行操作，

所以上述程序如果改为

```
int array[5]={1,4,3,5,2};
for(int i=0;i<5;i++)
    cout<<array[i]<<"";
cout<<endl;

memset(array,1,5*sizeof(int));//注意这里与上面的程序不同
for(int k=0;k<5;k++)
    cout<<array[k]<<"";
cout<<endl;
```

输出的结果就是：

```
14352
16843009168430091684300916843009168430009
```

为什么呢？

因为memset是以字节为单位就是对array指向的内存的4个字节进行赋值，字节，合一起就是

```
00000001000000010000000100000001
```

就等于16843009，就完成了对一个INT元素的赋值了。

所以用memset对非字符型数组赋初值是不可取的！

例如有一个结构体Some x，可以这样清零：

```
memset(&x,0,sizeof(Some));
```

如果是一个结构体的数组Some x[10]，可以这样：

```
memset(x,0,sizeof(Some)*10）；
```

memset函数详细说明

1。void *memset(void *s,int c,size_tn)

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

```
int main()
{
    char *s="GoldenGlobalView";
    clrscr();
    memset(s,'G',6);//貌似这里有点问题//这里没有问题，可以编译运行，楼主在这里将右括号和
//单步运行到这里会提示内存访问冲突
//肯定会访问冲突，s指向的是不可写空间。
    printf("%s",s);
    getchar();
    return 0;
}
```

【以上例子出现内存访问冲突应该是因为s被当做常量放入程序存储空间，如果修改为 char s[]="Golden Global View";则没有问题了。】

【应该是没有问题的，字符串指针一样可以，并不是只读内存，可以正常运行】

【此实例可以正常编译运行，并不像楼主说的需要char s[]】

【memset(s,'G',6)这样是存在内存访问冲突的，因为s为常量字符串，不能修改的】

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'memset(a,'\0',sizeof(a));

5。补充：一点技巧

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_struct 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(structsample_struct));
```

如果是数组：

```
structsample_struct TEST[10];
```

则

```
memset(TEST,0,sizeof(structsample_struct)*10）；
```

另外：

如果结构体中有数组的话还是需要对数组单独进行初始化处理的。

参考资料

1. memset API reference . www.cplusplus.com [引用日期2012-08-16] .

词条标签： 计算机学， 科技