# 1 关于元素个数为0的数组use心得

最近，工作时看了别人的代码，有个chance接触这个0数组的使用。代码中要实现类似这么个功能：  
内存中有一段缓存区buf，buf前8字节存俩个int型值，接着是一段字符串：  
char buf[100];  
memset(buf, 0, 100);  
int id = 9;  
int len = 10;  
char msg[] = "123456";  
memcpy(buf, &id, 4);  
memcpy(buf+4, &len, 4);  
memcpy(buf+8, msg, strlen(msg));  
另外，有个结构体定义如下：（别人定义的）  
struct ss\_t{  
    int s\_id;  
    int len;  
    char \*msg;  
};  
代码中，有类似下面的一段：  
struct ss\_t \*st;  
...  
st = (struct ss\_t \*)buf;  
将buf强制转换成struct ss\_t类型。  
然后对st操作：  
当对st->msg操作时，出错。  
    printf("%s\n", st->msg);     
显然，这边msg只是一个指针，指针的指向不知道，所以就出错了。我开始的解决办法是抛  
弃这个结构体直接对buf进行操作，改动颇大。后来，想想，能不能改动小点。记得之前听过别人说C语言的0数组，就试试了。把结构体的定义换成：  
struct ss\_t{  
    int s\_id;  
    int len;  
    char msg[0];  
};  
然后再编译运行，正常工作。其实，结构体对0数组时没有分配空间的，但是，相当于定义了一个数组，数组没有元素，但是，数组的首地址又是刚好在len之后。引用msg，就相当于对那块地址操作了。所以，问题顺利解决。  
此时，sizeof（struct ss\_t） = 8;  
另外，把其中的0去掉，定义成如下：  
struct ss\_t{  
    int s\_id;  
    int len;  
    char msg[];  
}

还可以使用const char \* msg;  
程序还是照常运行。  
此时，sizeof(struct ss\_t) = 8。  
可见，有没有0俩者一样，而且运行效果也一样。  
这里，msg相当于一个常量指针.不过， 读取st->msg:  
printf("%s", st->msg);输出123456ok,成功了  
我写的一个完整例子：  
#include   
#include   
#define INT\_LEN sizeof(int)  
struct ss\_t{  
    int s\_id;  
    int len;  
    char msg[];  
};  
int main(void)  
{  
    int id;  
    int len;  
    char msg[] = "123456";  
    char \*buf;  
    struct ss\_t \*st;  
    buf = (char \*)malloc(sizeof(char) \* 100);  
    if (buf == NULL) {  
        return 1;  
    }  
    memset(buf, 0, 100);  
    id = 65;  
    len = 66;  
    memcpy(buf, &id, INT\_LEN);  
    memcpy(buf + INT\_LEN, &len, INT\_LEN);  
    memcpy(buf + 2 \* INT\_LEN, msg, strlen(msg));    
    st = (struct ss\_t \*)buf;  
    printf("st->s\_id %d st->len %d\n", st->s\_id, st->len);  
    printf("st->msg %s\n", st->msg+1);      
    return 0;  
}

# 2 c语言printf打印格式：

%d 整型

%ld

%p 打印指针地址

%u：无符号整型

%x：16进制

%c 字符

%f float

%lf double