哦！数组还能这么用，学到了！

C语言与C++编程 2020-10-09

The following article is from 编程珠玑 Author 守望先生

编程珠玑

Linux，C语言，C++，数据结构与算法，计算机基础，数据库，工具，资源【...

> 作者：守望，Linux应用开发者，目前在公众号【编程珠玑】 分享Linux/C/C++/数据结构与算法/工具等原创技术文章和学习资源。

这个问题源于读者在阅读redis源码时的一个疑问。

先看下面的代码，对于包含动态字符串成员的两个结构体Test0和Test1占用空间分别是多少呢?

```
//来源：公众号【编程珠玑】
//作者：守望先生
#include<stdio.h>
struct Test0
{
    int a;
    int b;
    char *c;
};
struct Test1
{
    int a;
    int b;
    char c[];
};
int main(void)
{
    printf("sizeof(struct Test0) = %zd\n",sizeof(struct Test0));
    printf("sizeof(struct Test1) = %zd\n",sizeof(struct Test1));
    return 0;
}
```

很多读者一眼就能看出来，在64位系统上，编译为64位程序，其输出结果为：

```
16
8
```

对于Test0的结果是16，通常没有什么疑问，毕竟4（int）+4（int）+8（指针）= 16，但是对于后者的结构体占用空间为8字节，有的读者可能会有疑问。（关于字节对齐，参考《字节对齐，看这篇就懂了》）

柔性数组（**flexible array**）

实际上这是在C99中引入的柔性数组的特性。即结构体的最后一个成员，可以不完整类型**(**一种缺乏足够的信息去描述一个完整对象的类型**)**的数组，但它使得整个结构体的大小就像没有这个成员一样。但是呢，当用结构体通过这个名字访问这个成员时，就像访问一个普通数组成员一样。

如果数组最终一个元素都没有的话，那么访问这个数组将会是未定义行为了。

正如我们前面所看到的：

```
struct Test1
{
    int a;
    int b;
    char c[];
};
```

成员c是一个数组，但是并没有指定大小，使用sizeof计算Test1，其占用空间也仅仅是8字节。

有什么好处?

那么使用柔性数组有什么好处呢?

内存申请和释放

假设分别使用两种类型的结构体，存储16字节的字符数据，需要申请内存。对于struct Test0：

```
strcut Test0 *t0 = malloc(sizeof(struct Test0));//为结构体申请内存
t0->c = malloc(sizeof(char) * 16);//为成员指向的数据申请内存
```

而对于struct Test1：

```
strcut Test1 *t1 = malloc(sizeof(struct Test1) + sizeof(char) * 16);
```

看出区别了吗？前者需要两次内存申请，而后者只需要一次。前者地址不连续（两次malloc），后者地址连续。而你访问成员c的时候，只需要下面这样就可以：

t1->c，和普通成员无异。

要判断它们的地址是否连续也非常简单，只需要分别打印b和c的地址就可以了。

和内存释放类似，前面需要单独释放成员c申请的内存，而后者可以一起释放。

数据拷贝

正由于前面的差别，导致数据拷贝时，更有区别。
对于struct Test0：

```
//memcpy(t0copy,t0,sizeof(struct Test0));//不可，这样直接t0copy的c和t0的c指向同-

t0copy.a = t0.a;
t0copy.b = t0.b;
memcpy(t0copy.c,t0.c,sizeof(char)*16);
```

这里无法一次拷贝，因为它的成员c是一个指针类型，我们需要的是一份完整拷贝，因此必须拷贝它指向的内存。（参考《结构体成员赋值到底是深拷贝还是浅拷贝？》）

但是对于struct Test1：

```
memcpy(t0copy,t0,sizeof(strcut Test1) + sizeof(char) * 16);
```

在这里，由于柔性数组的内存，它的数据内容和结构体数据成员的地址是连续的，因此可以直接拷贝。

减少内存碎片

由于结构体的柔性数组和结构体成员的地址是连续的，即可一同申请内存，因此更大程度地避免了内存碎片。另外由于该成员本身不占结构体空间，因此，整体而言，比普通的数组成员占用空间要会稍微小点。

零长数组

与柔性数组功能类似，还有一个0长数组，不过它并不是标准中的，但是它可以实现类似的功能，使用方式如下：

```
struct Test1
{
    int a;
    int b;
    char c[0];
};
```

差别在于使得数组长度为0。但是由于它并非C标准中的，因此从可移植性考虑，不建议使用这种方式，除非你还无法使用C99。

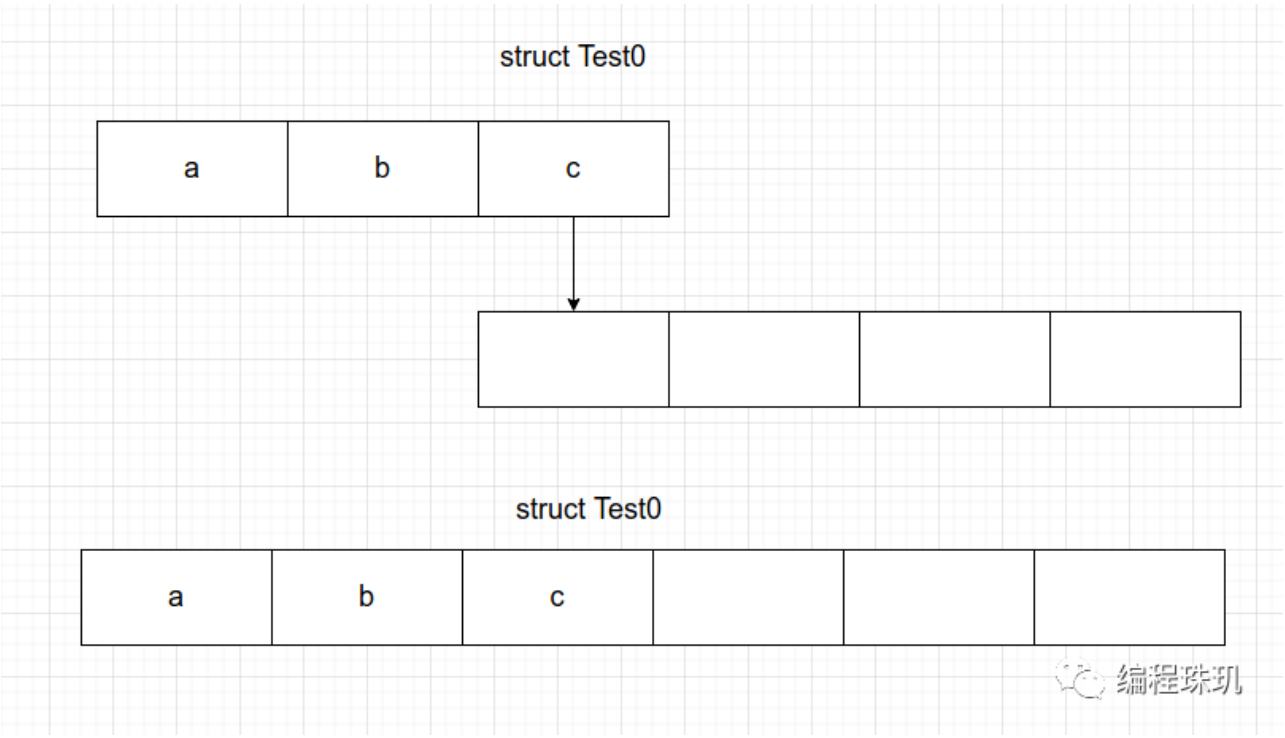
总结

柔性数组的使用：

- 位于结构体最后一个位置
- 不完整数组类型
- 不是唯一成员

最后，放张图，看差别：

普通和柔性数组



●输入m获取文章目录

C语言与C++编程

分享**C/C++**技术文章

喜欢此内容的人还喜欢

来看一道“简单的”C语言面试题

C语言与C++编程

阅读张一鸣，从他 10 年里的2000 多条微博

群响刘老板

请杨幂救场也没用，这季奇葩说难看在哪儿

万星人