High一下!

酷 壳 – COOLSHELL

享受编程和技术所带来的快乐 – Coding Your Ambition
(http://coolshell.cn/)

Search ...

C/C++返回内部静态成员的陷阱

2006年11月16日 (Http://Coolshell.cn/Articles/12192.Html) 陈皓 (Http://Coolshell.cn/Articles/Author/Haoel) 2,100 人阅读

在我们用C/C++开发的过程中，总是有一个问题会给我们带来苦恼。这个问题就是函数内和函数外代码需要通过一块内存来交互（比如，函数返回字符串），这个问题困扰和很多开发人员。如果你的内存是在函数内栈上分配的，那么这个内存会随着函数的返回而被弹栈释放，所以，你一定要返回一块函数外部还有效的内存。

这是一个让无数人困扰的问题。如果你一不小心，你就很有可能在这个上面犯错误。当然目前有很多解决方法，如果你熟悉一些标准库的话，你可以看到许多各式各样的解决方法。大体来说有下面几种：

1）在函数内部通过malloc或new在堆上分配内存，然后把这块内存返回（因为在堆上分配的内存是全局可见的）。这样带来的问题就是潜在的内存问题。因为，如果返回出去的内存不释放，那么就是memory Leak。或者是被多次释放，从而造成程序的crash。这两个问题都相当的严重，所以这种设计方法并不推荐。（在一些Windows API中，当你调用了一些API后，你必需也要调用他的某些API来释放这块内存）

2）让用户传入一块他自己的内存地址，而在函数中把要返回的内存放到这块内存中。这是一个目前普遍使用的方式。很多Windows API函数或是标准C函数都需要你传入一个buffer和这个buffer的长度。这种方式对我们来说应该是屡见不鲜了。这种方式的好处就是由函数外部的程序来维护这块内存，比较简显直观。但问题就是在使用上稍许有些麻烦。不过这种方式把犯错误的机率减到了最低。

3）第三种方式显得比较另类，他利用了static的特性，static的栈内存一旦分配，那这块内存不会随着函数的返回而释放，而且，它是全局可见的（只要你有这块内存的地址）。所以，有一些函数使用了static的这个特性，即不用使用堆上的内存，也不需要用户传入一个buffer和其长度。从而，使用得自己的函数长得很漂亮，也很容易使用。

这里，我想对第三个方法进行一些讨论。使用static内存这个方法看似不错，但是它有让你想象不到的陷阱。让我们来用一个实际发生的案例来举一个例子吧。

示例

有过socket编程经验的人一定知道一个函数叫：inet_ntoa，这个函数主要的功能是把一个数字型的IP地址转成字符串，这个函数的定义是这样的（注意它的返回值）：

```
char *inet_ntoa(struct in_addr in);
```

显然，这个函数不会分配堆上的内存，而他又没有让你传一下字符串的buffer进入，那么他一定使用“返回static char[]”这种方法。在我们继续我们的讨论之前，让我们先了解一下IP地址相关的知识，下面是inet_ntoa这个函数需要传入的参数：（也许你会很奇怪，只有一个member的struct还要放在struct中干什么？这应该是为了程序日后的扩展性的考虑）

```
struct in_addr {
unsigned long int s_addr;
}
```

对于IPV4来说，一个IP地址由四个8位的bit组成，其放在s_addr中，高位在后，这是为了方便网络传输。如果你得到的一个s_addr的整型值是：3776385196。那么，打开你的Windows计算器吧，看看它的二进制是什么？让我们从右到左，8位为一组（如下所示）。

```
11100001  00010111  00010000  10101100
```

再把每一组转成十进制，于是我们就得到：225 23 16 172， 于是IP地址就是 172.16.23.225。

好了，言归正传。我们有这样一个程序，想记录网络包的源地址和目地地址，于是，我们有如下的代码：

```
struct in_addr src, des;
........
........
fprintf(fp, "源IP地址<%s>/t目的IP地址<%s>/n", inet_ntoa(src),   inet_ntoa(des));
```

会发生什么样的结果呢？你会发现记录到文件中的源IP地址和目的IP地址完全一样。这是什么问题呢？于是你开始调试你的程序，你发现src.s_addr和des.s_addr根本不一样（如下所示）。可为什么输出到文件的源和目的都是一样的？难道说是inet_ntoa的bug？

```
src.s_addr = 3776385196;    //对应于172.16.23.225
des.s_addr = 1678184620;  //对应于172.16.7.100
```

原因就是inet_ntoa()“自作聪明”地把内部的static char[]返回了，而我们的程序正是踩中了这个陷阱。让我们来分析一下fprintf代码。在我们fprintf时，编译器先计算inet_ntoa(des)，于是其返回一个字符串的地址，然后程序再去求inet_ntoa(src)表达式，又得到一个字符串的地址。这两个字符串的地址都是inet_ntoa()中那个static char[]，显然是同一个地址，而第二次求src的IP时，这个值的des的IP地址内容必将被src的IP覆盖。所以，这两个表达式的字符串内存都是一样的了，此时，程序会调用fprintf把这两个字符串（其实是一个）输出到文件。所以，得到相同的结果也就不奇怪。

仔细看一下inet_ntoa的man，我们可以看到这句话：**The string is returned in a statically allocated buffer, which subsequent calls will overwrite.** 证实了我们的分析。

小结

让我们大家都扪心自问一下，我们在写程序的过程当中是否使用了这种方法？这是一个比较危险，容易出错的方法。这种陷阱让人防不胜防。想想，如果你有这样的程序：

```
if ( strcmp( inet_ntoa(ip1), inet_ntoa(ip2) )==0 ) {
.... ....
}
```

本想判断一下两个IP地址是否一样，却不料掉入了那个陷阱——让这个条件表达式永真。

这个事情告诉我们下面几个道理：

1）慎用这种方式的设计。返回函数内部的static内存有很大的陷阱。
2）如果一定要使用这种方式的话。你就必须严肃地告诉所有使用这个函数的人，千万不要在一个表达式中多次使用这个函数。而且，还要告诉他们，不copy函数返回的内存的内容，而只是保存返回的内存地址或是引用是没用的。不然的话，后果概不负责。
3）C/C++是很危险的世界，如果你不清楚他的话。还是回火星去吧。

附：看过Efftive C++的朋友一定知道其中有一个条款（item 23）：不要试图返回对象的引用。这个条款中也对是否返回函数内部的static变量进行了讨论。结果也是持否定态度的。

(全文完)

关注CoolShell微信公众账号可以在手机端搜索文章

(http://www.jiathis.com/share?uid=1541368)
(5 人打了分，平均分： 4.80)

C/C++语言 (Http://Coolshell.cn/Category/Proglanguage/Cplusplus), 编程语言 (Http://Coolshell.cn/Category/Proglanguage)
C++ (Http://Coolshell.cn/Tag/C)

相关文章

- 2009年09月19日 C++的std::string的“读时也拷贝”技术！ (http://coolshell.cn/articles/1443.html)
- 2014年04月21日 C语言的整型溢出问题 (http://coolshell.cn/articles/11466.html)
- 2009年04月17日 C语言下的错误处理的问题 (http://coolshell.cn/articles/551.html)
- 2011年11月01日 深入理解C语言 (http://coolshell.cn/articles/5761.html)
- 2012年09月20日 C/C++语言中闭包的探究及比较 (http://coolshell.cn/articles/8309.html)
- 2004年06月23日 C++ STL string的Copy-On-Write技术 (http://coolshell.cn/articles/12199.html)
- 2009年05月19日 谁说C语言很简单？ (http://coolshell.cn/articles/873.html)
- 2009年05月31日 C语言的谜题 (http://coolshell.cn/articles/945.html)

《C/C++返回内部静态成员的陷阱》的相关评论

canny38说道：

2015年02月08日 21:22 (http://coolshell.cn/articles/12192.html#comment-1657743)

唔，返回static变量的接口虽然方便，但是会有这个问题。
目前还有一种常见的方案，是返回一个带引用计数的内存块，默认计数为0，意味着下一帧这块内存被释放。

甘草说道：

2015年04月22日 18:31 (http://coolshell.cn/articles/12192.html#comment-1695194)
inet_ntoa实现真是坑人啊。

tree说道：

2015年06月07日 01:07 (http://coolshell.cn/articles/12192.html#comment-1718350)
使用localtime()时同样遇到这个问题

baixiangcpp说道：

2015年07月08日 00:00 (http://coolshell.cn/articles/12192.html#comment-1732644)
看你的博客比看书还有营养O(∩_∩)O

Pingback： 程序员技术练级攻略 | Zane's Blog (http://www.holahola.club/?p=171)

xmanxihua说道：

2017年01月01日 12:50 (http://coolshell.cn/articles/12192.html#comment-1909539)
static静态变量，还有线程安全问题。。。