

wangshubo1989的博客

一蓑烟雨任平生 也无风雨也无晴

RSS订阅

原 实战c++中的vector系列--正确释放vector的内存(clear(), swap(), shrink_to_fit())

2015年12月19日 21:18:19

阅读数：22019

关于vector已经写的差不多了，似乎要接近尾声了，从初始化到如何添加元素再到copy元素都有所涉及，是时候谈一谈内存的释放了。

是的，对于数据量很小的vector，完全没必要自己进行主动的释放，因为那样对程序的效率几乎没有影响。但是当vector中存入大量的数据后，并且都数据进行了一些操作，比如删除后，如果我们能积极主动的去释放内存，那么是非常明智的。

写到这里，应该明确了size和capacity的区别了。

现在介绍一个方法，**std::vector::clear()**

Removes all elements from the vector (which are destroyed), leaving the container with a size of 0.

看清楚了吗，英文中提到的是size=0，而非capacity。写程序验证一些：

```
1  #include<iostream>
2  #include<vector>
3  using namespace std;
4  int main()
5  {
6      vector<int> v;
7      v.push_back(1);
8      v.push_back(2);
9      v.push_back(3);
10     v.push_back(4);
11     v.push_back(5);
12
13     cout << "size:" << v.size() << endl;
14     cout << "capacity:" << v.capacity() << endl;
15
16     v.clear();
17     cout << "after clear size:" << v.size() << endl;
18     cout << "after clear capacity:" << v.capacity() << endl;
19     return 0;
20 }
21 //输出
22 size:5
23 capacity:6
24 after clear size:0
25 after clear capacity:6
```

看到了吗，clear后，size变为了0，capacity没有变化。再读一读clear的英文描述：

A reallocation is not guaranteed to happen, and the **vector capacity is not guaranteed** to change due to calling this function. A typical alternative that forces a reallocation is to use swap:

vector().swap(x); // clear x reallocating

所以这个时候swap该出厂了。

std::vector::swap

Exchanges the content of the container by the content of x, which is another vector object of the same type. Sizes may differ.

Notice that a non-member function exists with the same name, swap, overloading that algorithm with an optimization that behaves like this member function.

直接看看使用：

```
1  #include <iostream>
2  #include <vector>
3
4  int main()
5  {
6      std::vector<int> foo;
7      foo.push_back(1);
8      foo.push_back(2);
9      foo.push_back(3);
10     foo.push_back(4);
11     foo.push_back(5);
12
13     std::vector<int> bar;
14     bar.push_back(1);
15     bar.push_back(2);
16
17
18     std::cout << "foo size:" << foo.size() << std::endl;
19     std::cout << "foo capacity:" << foo.capacity() << std::endl;
20
21     std::cout << "bar size:" << bar.size() << std::endl;
22     std::cout << "bar capacity:" << bar.capacity() << std::endl;
23     foo.swap(bar);
24
25     std::cout << "after swap foo size:" << foo.size() << std::endl;
26     std::cout << "after swap foo capacity:" << foo.capacity() << std::endl;
27
28     std::cout << "after swap bar size:" << bar.size() << std::endl;
29     std::cout << "after swap bar capacity:" << bar.capacity() << std::endl;
30
31     return 0;
32 }
33 //输出:
34 foo size:5
35 foo capacity:6
36 bar size:2
37 bar capacity:2
38 after swap foo size:2
39 after swap foo capacity:2
40 after swap bar size:5
41 after swap bar capacity:6
```

看到了吗，swap之后，不仅仅是size变化了，capacity也是变化了。那么于是就把swap替代clear了：

```
1  #include<iostream>
2  #include<vector>
3  using namespace std;
4  int main()
5  {
6      vector<int> v;
7      v.push_back(1);
8      v.push_back(2);
9      v.push_back(3);
10     v.push_back(4);
11     v.push_back(5);
12
13     cout << "size:" << v.size() << endl;
```

```
14     cout << "capacity:" << v.capacity() << endl;
15
16     vector<int>().swap(v);
17     cout << "after swap size:" << v.size() << endl;
18     cout << "after swap capacity:" << v.capacity() << endl;
19     return 0;
20 }
21 //输出:
22 size:5
23 capacity:6
24 after swap size:0
25 after swap capacity:0
```

还记得上篇博客的shrink_to_fit()吗，如果clear后在调用shrink_to_fit()不一样可以吗？

```
1  #include<iostream>
2  #include<vector>
3  using namespace std;
4  int main()
5  {
6      vector<int> v;
7      v.push_back(1);
8      v.push_back(2);
9      v.push_back(3);
10     v.push_back(4);
11     v.push_back(5);
12
13     cout << "size:" << v.size() << endl;
14     cout << "capacity:" << v.capacity() << endl;
15
16     v.clear();
17     v.shrink_to_fit();
18     cout << "after swap size:" << v.size() << endl;
19     cout << "after swap capacity:" << v.capacity() << endl;
20     return 0;
21 }
22 //输出:
23 size:5
24 capacity:6
25 after swap size:0
26 after swap capacity:0
```

所以 不用以为只有swap替代clear才能正确释放vector的内存，C++11推出了shrink_to_fit方法，也可以达到目的。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/wangshubo1989/article/details/50359750>

文章标签：[vector](#) [c++](#)

个人分类：[C++](#)

所属专栏：[实战c++中的vector系列](#)





想对作者说点什么？

我来说两句



一蓁烟雨1989 2017-01-23 14:16:44 #2楼

Remarks: shrink_to_fit is a non-binding request to reduce capacity() to size(). [Note: The request is non-binding to allow latitude for implementation-specific optimizations. —end note]



shiter 2016-01-06 21:55:52 #1楼

[查看回复\(1\)](#)

请问一下，clear后，size变为了0，capacity没有变化，是代表了什么呢？谢谢分享

上一页

1

下一页

上一页

1

下一页

vector::clear（）方法的使用细节

原型: #include void clear(): 函数clear()删除储存在vector中的所有元素.



rl529014 2016-02-25 21:03:44 阅读数：17533

C++ 如何快速清空vector以及释放vector内存？

平时我们在写代码时候，有思考过要主动去释放vector的内存吗？ 1、对于数据量不大的vector，没有必要自己主动释放vector，一切都交给操作系统。 2、但是对于大量数据的vector，在vec...



hellokandy 2017-11-10 15:19:41 阅读数：1395

2018减肥新方法：每天坚持10分钟，甩掉大肚腩！

眼见科技 · 顶新



原来每天坚持这样做可以轻松消灭灰指甲

凯唐商贸 · 顶新



C++学习之路（vector::clear和vector::erase的区别）

vector::clear()函数的作用是清空容器中的内容，但并不回收内存，但你可以通过一下...



a18826408828 2014-07-17 21:54:34 阅读数：8861

vector的clear()和swap()比较

假设有若干对象存于一个 vector 中： class Widget; vector<Widget> vw; 后来由于某些原因，从该容器中删除了若干对象（参考erase-remove ...



u011450537 2015-01-03 20:21:17 阅读数：2028

使用STL vector的几种清空容器（删除）办法


转载自:http://blog.csdn.net/metalkittie/article/details/3115750vector vec1nt; for (int i=0;i



u012580994 2015-02-25 02:49:35 阅读数：34902

vector用完后需要释放嘛？

如果map,vector中存放了指针，指向手动分配的内存区域，则map,vector生命周期结束时，需要手动释放该内存区。map,vector的析构中带有垃圾回收机制，不需手动清空。记得，手动分配，才...

 PengPengBlog 2016-11-08 14:26:38 阅读数：2003

灰指甲怎么办?教你一个民间祛除灰指甲方法

凯唐商贸 · 顶新



男人补肾方法！教你1个方法，做回雄风男人！

星暹 · 顶新



vector 避免内存频繁分配释放与手动释放vector内存

1.避免频繁重分配 关于STL容器，最令人称赞的特性之一就是只要不超过它们的最大大小，它们就可以自动增长到足以容纳你放进去的数据。（要知道这个最大值，只要调用名叫max_size的成员函数。） ...

 cws1214 2015-08-25 21:27:57 阅读数：6537

vector 释放内存 swap

http://blog.csdn.net/jerryjbiao/article/details/7389233 相信大家看到swap这个词都一定不会感到陌生，甚至会有这样想法：这不...

 sunmenggmail 2013-02-23 16:51:54 阅读数：36821


vector动态二维数组（容器的容器）占用内存分析

用vector创建二维动态数组，也就是用容器的容器来创建，分析它是怎么使用内存的。...

 KangRoger 2014-08-18 21:21:54 阅读数：10663

vector内存释放机制

vector 中的内建有内存管理，当 vector 离开它的生存期的时候，它的析构函数会把 vector 中的元素销毁，并释放它们所占用的空间，所以用 vector 一般不...

 u014774781 2015-09-03 20:40:24 阅读数：3270

没有更多推荐了，[返回首页](#)

个人资料



一蓑烟雨1989

 博客专家

关注

原创	粉丝	喜欢	评论
571	1505	1183	846

等级：

访问：531万+

积分：4万+

排名：92

勋章：



最新文章

人与人为什么会疏远，这是我见过的最好答案

风雪夜归人

在简历上写了“精通 C++”后.....

Go实战--Design Patterns in Golang 之工厂模式(简单工厂、工厂方法、抽象工厂)

Go实战--Design Patterns in Golang 之单利模式(Singleton)

博主专栏

C++你所不知道的事儿

阅读量：152554 篇

浅析C++11新特性

阅读量：16396231 篇

那些年躲过的坑儿

阅读量：20749132 篇

一起学libcef

阅读量：1437997 篇

实战c++中的string系列

展开

个人分类

WPF

20篇

go

128篇

C++	258篇
MFC	17篇
C#	36篇
展开	

- 苹果电脑的
- python培训机构



笔记本电脑排行榜



联系我们



请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服  客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

 百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

联系我们



请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服  客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

 百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心