

zjufirefly

目录视图


摘要视图

RSS 订阅


管理博客

写新文章

个人资料



zjufirefly



访问：6425次

积分：464

等级：BLOG > 2

排名：千里之外

原创：42篇 转载：4篇

译文：0篇 评论：1条

文章搜索

文章分类

C++ (14)

其他 (5)

链接 (13)

linux (13)

shell (7)

操作 (11)

文章存档

2014年12月 (8)

2014年11月 (7)

2014年10月 (8)

2014年09月 (6)

2014年08月 (4)

展开

阅读排行

git操作命令 (327)

blktrace (287)

mysql常用命令 (280)

python (270)

cgroups (262)

linux下如何配置openvpn (260)

2015年4月微软MVP申请 《京东技术解密》有奖试读，礼品大放送 “我的2014”年度征文活动火爆开启 CSDN 2014博客之星

使用隐式转换auto\_ptr导致程序崩溃原因

分类：C++ 2014-05-02 18:01 122人阅读 评论(0) 收藏 编辑 删除

《C++标准程序库》p40指出，auto\_ptr<>不允许你使用一般指针惯用的赋值初始化方式，你必须直接使用数值来完成初始化：

```
[cpp]
01. std::auto_ptr<ClassA> ptr1(new ClassA); // OK
02. std::auto_ptr<ClassA> ptr2 =new ClassA; // ERROR
```

备注：

下面两种情况实际上是有区别的：

X x;

Y y(x); // 显示转换（explicit conversion）

和：

X x

Y y = x; // 隐式转换（implicit conversion）

前者使用显示转换，以型别X构造型别Y的一个对象，后者使用隐式转换。

执行以下使用隐式转换auto\_ptr的代码将会导致程序崩溃，原因是什么？

```
[cpp]
01. #include "stdafx.h"
02. #include <iostream>
03. #include <string>
04. #include <utility>
05. #include <memory>
06.
07. using namespace std;
08.
09. class A
10. {
11. };
12.
13. int _tmain(int argc, _TCHAR* argv[])
14. {
15.     A* pa = new A();
16.     auto_ptr<A> api = pa;
17.
18.     return 0;
19. }
```

auto\_ptr模板类的声明及实现如下：

```
[cpp]
```

vim-ctags-taglist-netrw	(249)
性能测试命令字段解释	(213)
正则表达式	(211)
SystemTap	(195)

评论排行	
为什么需要auto_ptr_ref	(1)
valgrind内存检查	(0)
cgroups	(0)
linux下如何配置openvpn	(0)
git操作命令	(0)
thrift	(0)
Apache开源软件	(0)
vim-ctags-taglist-netrw	(0)
vim常用配置	(0)
shell按行读取字符串，并	(0)

推荐文章	
* 挣扎与彷徨——我的2014	
* 校招回忆录——小米篇	
* Android UI-自定义日历控件	
* 30岁程序员回顾人生、展望未来	
* 2014年终总结，我决定要实现的三个目标	
* Android 启动问题——黑屏 死机解决方法	

最新评论	
为什么需要auto_ptr_ref	
zjufirefly: explicit auto_ptr_ref(Tp1* __p): _M_ptr(__p){}...	

```
01. template<class _Ty>
02.     class auto_ptr
03.     { // wrap an object pointer to ensure destruction
04. public:
05.     typedef _Ty element_type;
06.
07.     explicit auto_ptr(_Ty * _Ptr = 0) _THROW0()
08.         : _Myptr(_Ptr)
09.     { // construct from object pointer
10.     }
11.
12.     auto_ptr(auto_ptr<_Ty>& _Right) _THROW0()
13.         : _Myptr(_Right.release())
14.     { // construct by assuming pointer from _Right auto_ptr
15.     }
16.
17.     auto_ptr(auto_ptr_ref<_Ty> _Right) _THROW0()
18.     { // construct by assuming pointer from _Right auto_ptr_ref
19.     _Ty **_Pptr = (_Ty **)_Right._Ref;
20.     _Ty * _Ptr = *_Pptr;
21.     *_Pptr = 0; // release old
22.     _Myptr = _Ptr; // reset this
23.     }
24. private:
25.     const _Ty * _Myptr; // the wrapped object pointer
26.     };
```

explicit auto\_ptr(\_Ty \* \_Ptr = 0) \_THROW0(), 以指针为入参的构造函数要求显示转换。  
auto\_ptr(auto\_ptr\_ref<\_Ty> \_Right) \_THROW0(), 以auto\_ptr\_ref为入参的构造函数可以进行隐式转换。

auto\_ptr<A> api = pa;  
由于无法调用explicit auto\_ptr(\_Ty \* \_Ptr = 0) \_THROW0()进行显示转换构造auto\_ptr对象，先以pa为入参调用auto\_ptr\_ref的构造函数构造auto\_ptr\_ref对象，再调用auto\_ptr(auto\_ptr\_ref<\_Ty> \_Right) \_THROW0()构造函数由auto\_ptr\_ref对象构造auto\_ptr对象。

```
[cpp] C {}
01. template<class _Ty>
02.     struct auto_ptr_ref
03.     { // proxy reference for auto_ptr copying
04.     auto_ptr_ref(void * _Right)
05.         : _Ref(_Right)
06.     { // construct from generic pointer to auto_ptr ptr
07.     }
08.
09.     void * _Ref; // generic pointer to auto_ptr ptr
10.     };
```

auto\_ptr\_ref中的\_Ref值为pa的值。

然后调用如下代码构造auto\_ptr。

```
[cpp] C {}
01. auto_ptr(auto_ptr_ref<_Ty> _Right) _THROW0()
02. { // construct by assuming pointer from _Right auto_ptr_ref
03. _Ty **_Pptr = (_Ty **)_Right._Ref;
04. _Ty * _Ptr = *_Pptr;
05. *_Pptr = 0; // release old
06. _Myptr = _Ptr; // reset this
07. }
```

auto\_ptr中的\_Myptr 值为\_Ref指针所指向内存处的4个字节所构成的值（如果指针占用4个字节），导致\_Myptr为野指针。当auto\_ptr析构时将delete \_Myptr，删除野指针，造成程序崩溃。  
以上造成程序崩溃代码为VS2005中的实现。使用C++标准程序库中自带代码则不会出现问题，只是多了一次转换造成性能下降。

下一篇 为什么需要auto\_ptr\_ref

主题推荐

namespace 指针 对象 标准 iostream

猜你在找

- C++内存管理
- C++内存管理
- C++内存管理
- 内存管理
- C++内存管理
- C++内存管理
- C++进阶从内存管理内存泄漏内存回收探讨C++内存管
- c++内存管理
- C++内存管理
- 内存管理二

- 1 java教程
- 2 博客搬家工具
- 3 java
- 4 java培训
- 5 挡风玻璃的价格
- 6 java下载
- 7 开炼机
- 8 小型喷码机价钱
- 9 全国商标查询
- 10 易拉罐分切机
- 11 黔西南交友网
- 12 焊工证查寻
- 13 猪猪棋牌游戏
- 14 下载捕鱼
- 15 大信整体橱柜
- 16 少儿奥数
- 17 ewin棋牌
- 18 酋长咖啡
- 19 盐雾试验设备
- 20 净化工程公司

查看评论

暂无评论

发表评论

用户名: zjufirefly

评论内容:

提交

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop
- AWS
- 移动游戏
- Java
- Android
- iOS
- Swift
- 智能硬件
- Docker
- OpenStack
- VPN
- Spark
- ERP
- IE10
- Eclipse
- CRM
- JavaScript
- 数据库
- Ubuntu
- NFC
- WAP
- jQuery
- BI
- HTML5
- Spring
- Apache
- .NET
- API
- HTML
- SDK
- IIS
- Fedora
- XML
- LBS
- Unity
- Splashtop
- UML
- components
- Windows Mobile
- Rails
- QEMU
- KDE
- Cassandra
- CloudStack
- FTC
- coremail
- OPhone
- CouchBase
- 云计算
- iOS6
- Rackspace
- Web App
- SpringSide
- Maemo
- Compuware
- 大数据
- aptech
- Perl
- Tornado
- Ruby
- Hibernate
- ThinkPHP
- HBase
- Pure
- Solr
- Angular
- Cloud Foundry
- Redis
- Scala
- Django
- Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持  
京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

