

如何限制对象只能建立在堆上或者栈上

原创 舒夜无痕 2013-09-25 10:32:57 8147 收藏 5

版权

分类专栏: C/C++

在C++中，类的对象建立分为两种，一种是静态建立，如A a；另一种是动态建立，如A* ptr=new A；这两种方式是有区别的。

静态建立一个类对象，是由编译器为对象在栈空间中分配内存，是通过直接移动栈顶指针，挪出适当的空间，然后在这片内存空间上调用构造函数形成一个栈对象。使用这种方法，直接调用类的构造函数。

动态建立类对象，是使用new运算符将对象建立在堆空间中。这个过程分为两步，第一步是执行operator new()函数，在堆空间中搜索合适的内存并进行分配；第二步是调用构造函数构造对象，初始化这片内存空间。这种方法，间接调用类的构造函数。

那么如何限制类对象只能在堆或者栈上建立呢？下面分别进行讨论。

1、只能建立在堆上

类对象只能建立在堆上，就是不能静态建立类对象，即不能直接调用类的构造函数。

容易想到将构造函数设为私有。在构造函数私有之后，无法在类外部调用构造函数来构造类对象，只能使用new运算符来建立对象。然而，前面已经说过，new运算符的执行过程分为两步，C++提供new运算符的重载，其实是只允许重载operator new()函数，而operator()函数用于分配内存，无法提供构造功能。因此，这种方法不可以。

当对象建立在栈上面时，是由编译器分配内存空间的，调用构造函数来构造栈对象。当对象使用完后，编译器会调用析构函数来释放栈对象所占的空间。编译器管理了对象的整个生命周期。如果编译器无法调用类的析构函数，情况会是怎样的呢？比如，类的析构函数是私有的，编译器无法调用析构函数来释放内存。所以，编译器在为类对象分配栈空间时，会先检查类的析构函数的访问性，其实不光是析构函数，只要是非静态的函数，编译器都会进行检查。如果类的析构函数是私有的，则编译器不会在栈空间上为类对象分配内存。

因此，将析构函数设为私有，类对象就无法建立在栈上了。代码如下：

```
1 class A
2 {
3 public:
4     A(){}
5     void destory(){delete this;}
6 private:
7     ~A(){}
8 };
```

试着使用A a;来建立对象，编译报错，提示析构函数无法访问。这样就只能使用new操作符来建立对象，构造函数是公有的，可以直接调用。类中必须提供一个destory函数，来进行内存空间的释放。类对象使用完成后，必须调用destory函数。

上述方法的一个缺点就是，无法解决继承问题。如果A作为其它类的基类，则析构函数通常要设为virtual，然后在子类重写，以实现多态。因此析构函数不能设为private。还好C++提供了第三种访问控制，protected。将析构函数设为protected可以有效解决这个问题，类外无法访问protected成员，子类则可以访问。

另一个问题是，类的使用很不方便，使用new建立对象，却使用destory函数释放对象，而不是使用delete。（使用delete会报错，因为delete对象的指针，会调用对象的析构函数，而析构函数类外不可访问）这种使用方式比较怪异。为了统一，可以将构造函数设为protected，然后提供一个public的static函数来完成构造，这样不使用new，而是使用一个函数来构造，使用一个函数来析构。代码如下，类似于单例模式：

点赞 7 评论 4 分享 收藏 5 手机看 打赏 ... 关注

```
1 | class A
   | 2 | {
3 | protected:
4 |     A(){}
5 |     ~A(){}
6 | public:
7 |     static A* create()
8 |     {
9 |         return new A();
10 |    }
11 |    void destory()
12 |    {
13 |        delete this;
14 |    }
15 |};
```

这样，调用create()函数在堆上创建类A对象，调用destory()函数释放内存。

2、只能建立在栈上

只有使用new运算符，对象才会建立在堆上，因此，只要禁用new运算符就可以实现类对象只能建立在栈上。

将operator new()设为私有即可。代码如下：

```
1 | class A
2 | {
3 | private:
4 |     void* operator new(size_t t){} // 注意函数的第一个参数和返回值都是固定的
5 |     void operator delete(void* ptr){} // 重载了new就需要重载delete
6 | public:
7 |     A(){}
8 |     ~A(){}
9 |};
```

参考：<http://blog.csdn.net/g5dsk/article/details/4775144>

——The End——

©2020 CSDN 皮肤主题: 大白 设计师: CSDN官方博客 返回首页

关于我们 招聘 广告服务 网站地图 kefu@csdn.net 客服论坛 400-660-0108 QQ客服 (8:30-22:00)
公安备案号 11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 版权与免责声明 版权申诉 网络110报警服务
中国互联网举报中心 家长监护 版权申诉 北京互联网违法和不良信息举报中心 ©1999-2020 北京创新乐知网络技术有限公司