原创 Qt 内存管理机制

2011-03-26 10:50:43

标签：qt 内存 休闲 职场

这篇文章首先发布于我的主页 http://www.devbean.info，以后也会直接发布在那里。现在有 Flex 4 的一篇和 《从 C++ 到 Objective-C》系列，感谢大家支持！

强类型语言在创建对象时总会显式或隐式地包含对象的类型信息。也就是说，强类型语言在分配对象内存空间时，总会关联上对象的类型。相比之下，弱类型 语言则不会这样做。在分配了内存空间之后，有两种方法释放空间：手工释放，或者是使用垃圾收集器。C++ 要求开发者手工释放内存空间。这样做的好处是，开发者对内存有完全的控制能力，知道什么时候释放比较合适。Java 则使用垃圾收集器。它在后台会有一个线程根据一定的算法不停地查看哪些对象已经不被使用，可以被回收。这样做则可以将开发者从底层实现中解放出来，只需关 注于业务逻辑。

本文关注于 Qt 的内存管理，这里会使用 Qt 的机制，来实现一个简单的垃圾回收器。

C++ 内存管理机制

C++ 要求开发者自己管理内存。有三种策略：

1. 让创建的对象自己 delete 自己的子对象（这里所说的子对象，是指对象的属性，而不是子类，以下类似）；
2. 让最后一个对象处理 delete；
3. 不管内存。

最后一种通常成为“内存泄漏”，被认为是一种 bug。所以，我们现在就是要选出前面两种哪一种更合适一些。有时候，delete 创建的对象要比 delete 它的所有子对象简单得多；有时候，找出最后一个对象也是相当困难的。

热门文章

Qt学习之路(17)：Qt标准..

Qt学习之路(2)：Hello, w..

Qt学习之路(1)：前言

Qt学习之路(50)：QString

Qt学习之路(7)：创建一个..

Qt学习之路(8)：创建一个..

Qt学习之路(15)：Qt标准..

深入Java单例模式

搜索BLOG文章

搜 索

最近访客

10552..　72974..　15192..

jim003　qinzy212　61940..

sucha..　肇庆..　一心永成

a9341..　loserfly　gaoku..

最新评论

飘雪夏娜样：书上说的都说了，书上没有讲清楚的..

Qt 内存管理机制

Qt 在内部能够维护对象的层次结构。对于可视元素，这种层次结构就是子组件与父组件的关系；对于非可视元素，则是一个对象与另一个对象的从属关系。在 Qt 中，删除父对象会将其子对象一起删除。这有助于减少 90% 的内存问题，形成一种类似垃圾回收的机制。

QPointer

QPointer 是一个模板类。它很类似一个普通的指针，不同之处在于，QPointer 可以监视动态分配空间的对象，并且在对象被 delete 的时候及时更新。

```
01.  // QPointer 表现类似普通指针
02.  QDate *mydate = new QDate(QDate::currentDate());
03.  QPointer mypointer = mydata;
04.  mydate->year();     // -> 2005
05.  mypointer->year(); // -> 2005
06.
07.  // 当对象 delete 之后，QPointer 会有不同的表现
08.  delete mydate;
09.
10.  if(mydate == NULL)
11.      printf("clean pointer");
12.  else
13.      printf("dangling pointer");
14.  // 输出 dangling pointer
15.
16.  if(mypointer.isNull())
17.      printf("clean pointer");
18.  else
19.      printf("dangling pointer");
20.  // 输出 clean pointer
```

注意上面的代码。一个原始指针 delete 之后，其值不会被设置为 NULL，因此会成为野指针。但是，QPionter 没有这个问题。

QObjectCleanupHandler

Qt 对象清理器是实现自动垃圾回收的很重要的一部分。它可以注册很多子对象，并在自己删除的时候自动删除所有子对象。同时，它也可以识别出是否有子对象被删 除，从而将其从它的子对象列表中删除。这个类可以用于不在同一层次中的类的清理操作，例如，当按钮按下时需要关闭很多窗口，由于窗口的 parent 属性不可能设置为别的窗口的 button，此时使用这个类就会相当方便。

```
01.  // 创建实例
02.  QObjectCleanupHandler *cleaner = new QObjectCleanupHandler;
03.  // 创建窗口
04.  QPushButton *w = new QPushButton("Remove Me");
05.  w->show();
06.  // 注册第一个按钮
07.  cleaner->add(w);
08.  // 如果第一个按钮点击之后，删除自身
09.  connect(w, SIGNAL(clicked()), w, SLOT(deleteLater()));
10.  // 创建第二个按钮，注意，这个按钮没有任何动作
11.  w = new QPushButton("Nothing");
12.  cleaner->add(w);
13.  w->show();
14.  // 创建第三个按钮，删除所有
15.  w = new QPushButton("Remove All");
16.  cleaner->add(w);
17.  connect(w, SIGNAL(clicked()), cleaner, SLOT(deleteLater()));
18.  w->show();
```

在上面的代码中，创建了三个仅有一个按钮的窗口。第一个按钮点击后，会删除掉自己（通过 deleteLater() 槽），此时，cleaner 会自动将其从自己的列表中清除。第三个按钮点击后会删除 cleaner，这样做会同时删除掉所有未关闭的窗口。

Qt 垃圾收集

aguai1617：请问有没有从应用程序拖动文件到本..

wx275271279：“这里其实是一个冗余的操作，因为..

wx275271279：另外还有一点，在一些必须精确操作..

wx275271279：讲的深入透彻，楼主厉害，十分感谢

51CTO推荐博文 更多>>

- 运维角度浅谈MySQL数据库优化
- Oracle Study之--AMD CPU安装Orac..
- svn利用钩子脚本功能实现代码同步..
- 【UNITY3D 游戏开发之八】Unity编..
- PL/SQL中如何让程序每隔几秒插入..
- Keepalived+LVS+MariaDB Galera C..
- Ubuntu14.04快速搭建SVN服务器及..
- 整理的部分Java和C#不同点
- Apache Httpd服务器之虚拟机详解
- 将Uhost上的MySQL迁移到UDB
- mysql分布式中间件cobar

友情链接

- 石榴石网
- 趣闻网
- 汉堡店加盟
- 奶茶店加盟
- IT精品课程
- 我的主页
- PicWorks
- Qt 文档翻译
- Exchange 2010
- 51CTO博客开发

随着对象变得越来越复杂，很多地方都要使用这个对象的时候，什么时候作 delete 操作很难决定。好在 Qt 对所有继承自 QObject 的类都有很好的垃圾收集机制。垃圾收集有很多种实现方法，最简单的是引用计数，还有一种是保存所有对象。下面我们将详细讲解这两种实现方法。

引用计数

应用计数是最简单的垃圾回收实现：每创建一个对象，计数器加 1，每删除一个则减 1。

```
01. class CountedObject
02. {
03. public:
04.     CountedObject()
05.     {
06.         ctr=0;
07.     }
08.
09.     void attach()
10.     {
11.         ctr++;
12.     }
13.
14.     void detach()
15.     {
16.         ctr--;
17.         if(ctr <= 0)
18.             delete this;
19.     }
20. private:
21.     int ctr;
22. };
```

每一个子对象在创建之后都应该调用 attach() 函数，使计数器加 1，删除的时候则应该调用 detach() 更新计数器。不过，这个类很原始，没有使用 Qt 方便的机制。下面我们给出一个 Qt 版本的实现：

```
01. class CountedObject : public QObject
02. {
03.     Q_OBJECT
04. public:
05.     CountedObject()
06.     {
07.         ctr=0;
08.     }
09.
10.     void attach(QObject *obj)
11.     {
12.         ctr++;
13.         connect(obj, SIGNAL(destroyed(QObject*)), SLOT(detach()));
14.     }
15.
16. public slots:
17.     void detach()
18.     {
19.         ctr--;
20.         if(ctr <= 0)
21.             delete this;
22.     }
23.
24. private:
25.     int ctr;
26. };
```

我们利用 Qt 的信号槽机制，在对象销毁的时候自动减少计数器的值。但是，我们的实现并不能防止对象创建的时候调用了两次 attach()。

记录所有者

更合适的实现是，不仅仅记住有几个对象持有引用，而且要记住是哪些对象。例如：

```
01. class CountedObject : public QObject
02. {
03. public:
04.     CountedObject()
05.     {
06.     }
07. 
08.     void attach(QObject *obj)
09.     {
10.         // 检查所有者
11.         if(obj == 0)
12.             return;
13.         // 检查是否已经添加过
14.         if(owners.contains(obj))
15.             return;
16.         // 注册
17.         owners.append(obj);
18.         connect(obj, SIGNAL(destroyed(QObject*)), SLOT(detach(QObject*)));
19.     }
20. 
21. public slots:
22.     void detach(QObject *obj)
23.     {
24.         // 删除
25.         owners.removeAll(obj);
26.         // 如果最后一个对象也被 delete，删除自身
27.         if(owners.size() == 0)
28.             delete this;
29.     }
30. 
31. private:
32.     QList owners;
33. };
```

现在我们的实现已经可以做到防止一个对象多次调用 attach() 和 detach() 了。然而，还有一个问题是，我们不能保证对象一定会调用 attach() 函数进行注册。毕竟，这不是 C++ 内置机制。有一个解决方案是，重定义 new 运算符（这一实现同样很复杂，不过可以避免出现有对象不调用 attach() 注册的情况）。

本文来自 DevBean's World：http://www.devbean.info。

转载时请标明文章原始出处：http://www.devbean.info/2011/03/qt_memory_management/。

上一篇 从 C++ 到 Objective-C 下一篇 Qt 学习之路 2

相关文章

- JVM内存模型以及垃圾收集策略解析【续】
- DDR内存和DDRII内存的区别
- 由函数调用约定引起的问题
- JVM内存调优
- DDR2与DDR的区别
- Qt学习之路(tip): Qt容器和算法拾遗
- 节约内存小技巧
- 火狐请绕了我的内存吧！！！

文章评论

[1楼] hyy1988 回复

2011-04-07 15:39:17

Qt是C＋＋方面 吗？

[2楼]楼主 FinderCheng 回复

2011-04-11 08:48:41

回复 hyy1988:[1楼]
Qt 是 C++ 的一个库，包含界面、XML、network 等等，基本可以提供一个完整的解决方案。

[3楼] uestc001 回复

2011-05-10 17:49:29

谢谢lz，一直以来，在你的不bolg学习不少东西

[4楼] [匿名]xdskc 回复

2011-06-15 10:42:56

支持豆子！希望看到更多更精彩的文章！

[5楼] [匿名]轨迹16 回复

2013-05-14 11:25:10

写得真的很好，由于这个博客，有点喜欢上了qt

[6楼] D_clock 2013-05-25 09:38:37 回复

博主，要是能图文并茂那就更好了！

发表评论 51CTO学院2周年庆，分享故事赢大礼包

昵　　称: 登录 快速注册

验证码: 请点击后输入验证码 博客过2级，无需填写验证码

内　　容: