QCore/Library 实现文档

李文超

前言

QCore/Library 是一套类 STL 的类库，它在标准库的范围内删去了不常用的 heap、deque 等结构（至少我是不常用的）。并为一些容器提供了一些特殊的接口，比如 vector 中的 push_back_unique、add 和 add_unique 等。

Library 主要分为六部分，内存调试相关、容器、算法、正则、IO 和 Graphic，每个模块都有各自的分工，他们之间的耦合度极低，几乎每个模块都可以拆出来独立使用，下面来分别介绍各个模块。

内存调试

我们知道，在 C/C++中内存相关的东西是极难控制的，在使用不当时可能造成各种错误，轻则内存泄漏，重则程序崩溃。所以，在生产环境中我们必须通过一个有效的手段来管理好内存。当然，在小块内存频繁 new、delete 的过程中也会产生大量的内存碎片，从而导致可用内存数量越来越少。应此我们设计了一个内存池来控制小块内存的频繁 new、delete，以及做到对内存泄漏的检测。

在内存池的设计之初，我只是简单的设计出了可以使用的 MemoryPool 的最简版本，它包含一个大块内存的 free_list 和每个小块内存的 chunk_list，当时足以应付大部分的需求，而且最初用的是 Visual Leak Detector 来检测内存泄漏。但随着时间的推移，想要自己检测内存泄漏的欲望越来越强烈，之后便有了一个 use_list 来保存内存块的释放情况。当时完成了这个 patch 之后，兴奋的跑了一下 TestCase，然后的结果我想大家应该知道了，一路的飘红，到处是内存泄漏。

经过一天的调试，实在无法容忍的情况下，我翻阅了 MSDN，查到了 dbghelp.dll 中可以通过许多函数来获取调用堆栈，于是在此之下便生产出了 CallStack 模块。有了它之后你就可以在任意地方保存当前的调用堆栈了，真是十分方便。当然直到现在，它还只支持在 Windows 下调用堆栈的获取（稍后我会翻阅资料，实现一个 like unix 的版本，如果可能的话）。

这里不过多的描述实现的细节，具体可以看 http://www.cppblog.com/lwch/archive/2012/07/14/183420.html 和 http://www.cppblog.com/lwch/archive/2013/01/19/197415.html 两篇文章。

容器

结束

2013.4.23 第一次编写