C++内存泄露检查的几个方法

在 Linux 平台上 有 valgrind 可以非常方便的帮助我们定位内存泄漏，因为 Linux 在开发领域的使用场景大多是跑服务器，再加上它的开源属性，相对而言，处理问题容 易形成“统一”的标准。而在 Windows 平台，服务器和客户端开发人员惯用的调试方法有很大不同。

一、前言

在 Linux 平台上 有 valgrind 可以非常方便的帮助我们定位内存泄漏，因为 Linux 在开发领域的使用场景大多是跑服务器，再加上它的开源属性，相对而言，处理问题容 易形成“统一”的标准。而在 Windows 平台，服务器和客户端开发人员惯用的调试方法有很大不同。下面结合我的实际经验，整理下常见定位内存泄漏的方 法。

注意：我们的分析前提是 Release 版本，因为在 Debug 环境下，通过 VLD 这个库或者 CRT 库本身的内存泄漏检测函数能够分析出内存泄漏，相对而言比较简单。而服务器有很多问题需要在线上并发压力情况下才出现，因此讨论 Debug 版调试方法意义不大。

二、对象计数

方法：在对象构造时计数++，析构时－，每隔一段时间打印对象的数量

优点：没有性能开销，几乎不占用额外内存。定位结果精确。

缺点：侵入式方法，需修改现有代码，而且对于第三方库、STL 容器、脚本泄漏等因无法修改代码而无法定位。

三、重载 new 和 delete

方法：重载 new/delete，记录分配点（甚至是调用堆栈），定期打印。

优点：没有看出

缺点：侵入式方法，需将头文件加入到大量源文件的头部，以确保重载的宏能够覆盖所有的 new/delete。记录分配点需要加锁（如果你的程序是多线程），而且记录分配要占用大量内存（也是占用的程序内存）。

四、Hook Windows 系统 API

方法：使用微软的 detours 库，hook 分配内存的系统 Api：HeapAlloc/HeapRealloc/HeapFree（new/malloc 的底层调用）,记录分配点，定期打印。

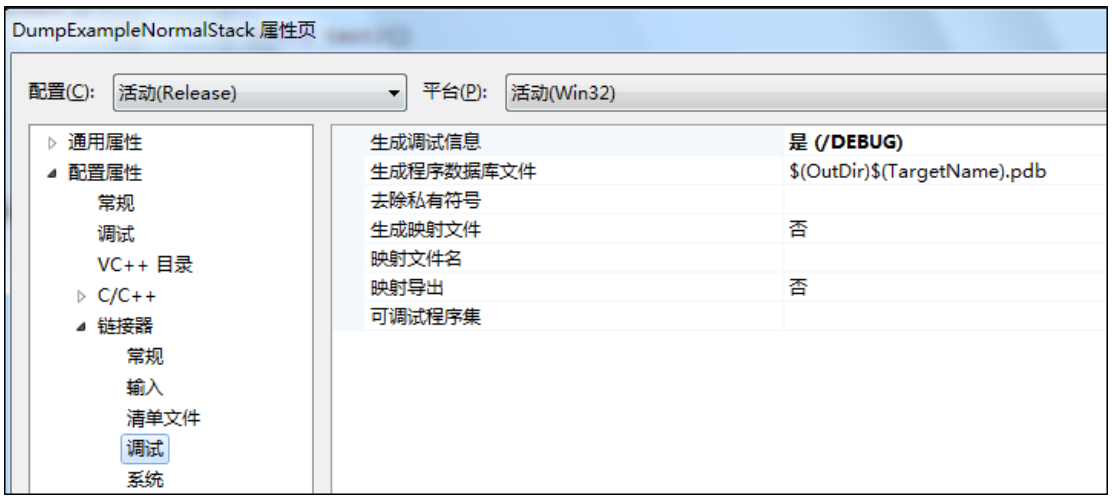
优点：非侵入式方法，无需修改现有文件（hook api 后，分配和释放走到自己的钩子函数中），检查全面，对第三方库、脚本库等等都能统计到。

缺点：记录内存需要占用大量内存，而且多线程环境需要加锁。

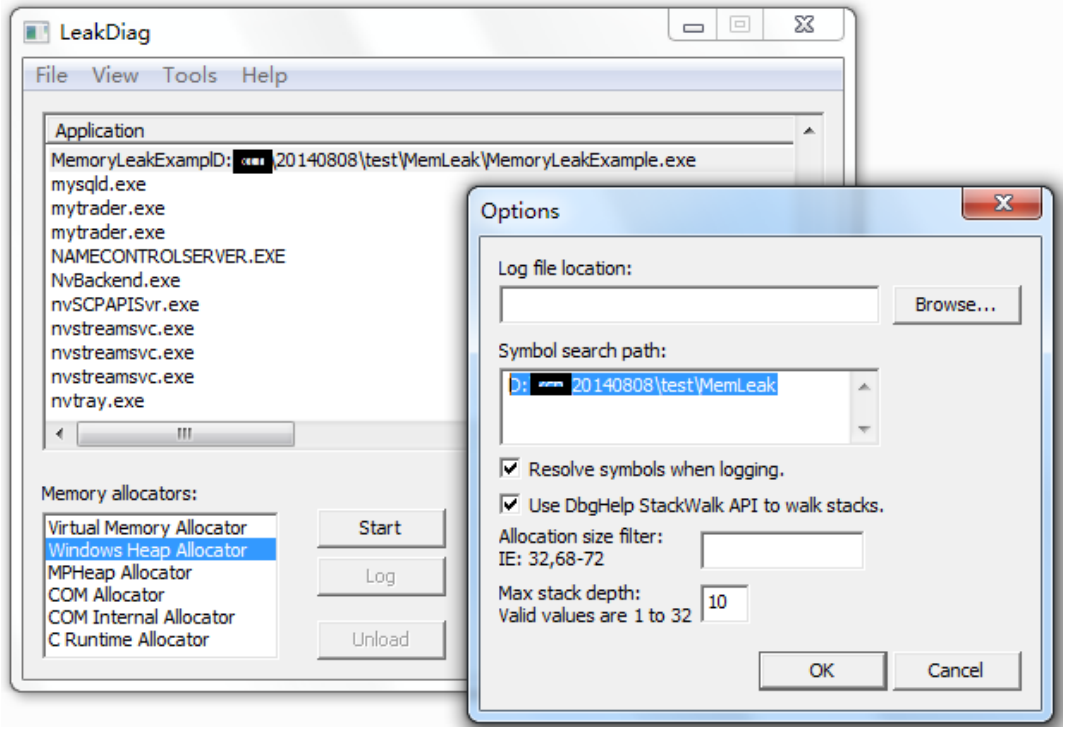
五、使用 DiagLeak 检测

微软出品的内存泄漏分析工具，原理同 hookapi 方式。配合 LDGraph 可视化展示内存分配数据，更方便查找泄漏。

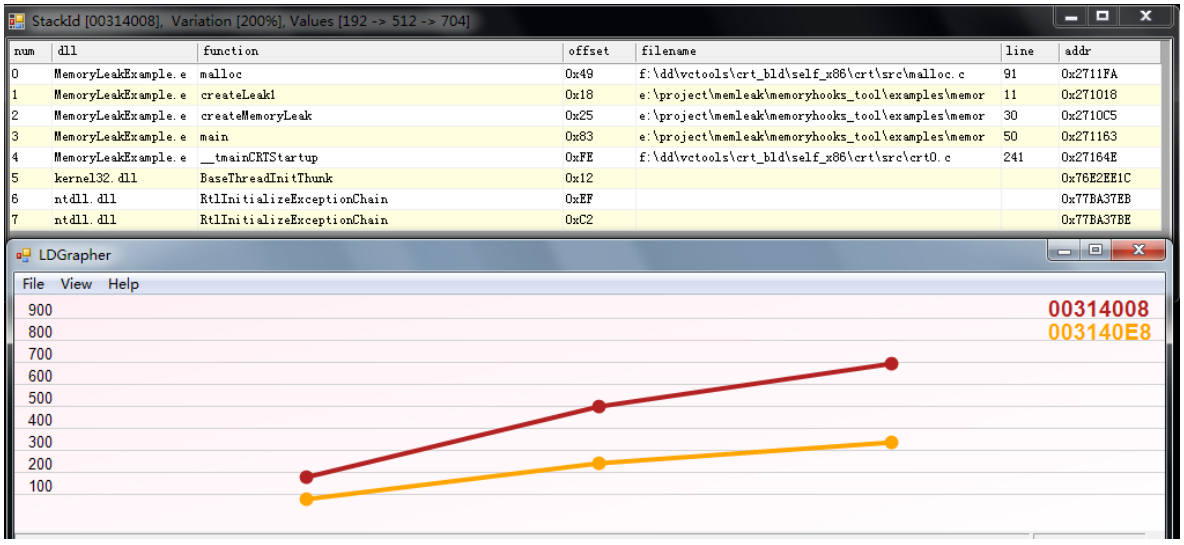
1.在 IDE 工程选项里面配置 Release 版本也生成调试信息，发布时，将 pdb 文件和 exe 文件一起发布。



2.程序运行后，打开 LeakDiag，设置 Symbol path



3.定期 Log 下目标进程的内存分配情况，通过 LDGraph 打印分配增长情况，来发现内存泄漏。



优点：同 hookapi 方法，非侵入式修改，无需做任何代码改动。跟踪全面。可视化分析堆栈一览无余！

缺点：对性能有影响，hook 分配加锁，遍历堆栈。但是不会占用目标进程的自身内存。

六、总结

对于线上生产环境，建议大对象用计数来判断，定位快速准确，几乎无性能开销。在对外测试阶段，使用 LeakDiag 辅助分析，因为此时并发压力还不 是太大，性能开销还是可以承受。

在线上大规模应用阶段，通过 HookApi 的方法，结合 GM 指令控制部分时间段的检测，这样可以把对玩家的影响（服务器性 能下降导致延迟）降到最低。