# **项目内存泄漏分析报告**

【摘要】

本报告采用VMMap内存分析工具、VLD内存泄露分析工具、UMDH内存占用分析工具以及VS内置诊断工具，对文件监听工具项目的内存占用情况、内存泄露问题以及内存优化方案进行说明。报告中将对各个分析工具的使用进行说明，分析以降低特定功能的内存使用量、排查内存占用异常上涨为目的，通过中断分析调用栈、记录内存分配比较差异的方法来进行。

【关键词】

内存泄漏、内存优化

**一、项目内存占比分析**

VMMap是对进程所用虚拟、物理内存进行分析的工具，可以统计进程所使用的虚拟、物理内存并以条形图与文字的形式进行呈现。VMMap的过滤和刷新功能可以及时观察进程内存使用的来源以及占用情况。

运行SVG编辑器项目，并使用VMMap工具进行检测，可首先观察到如下界面，

图1.1

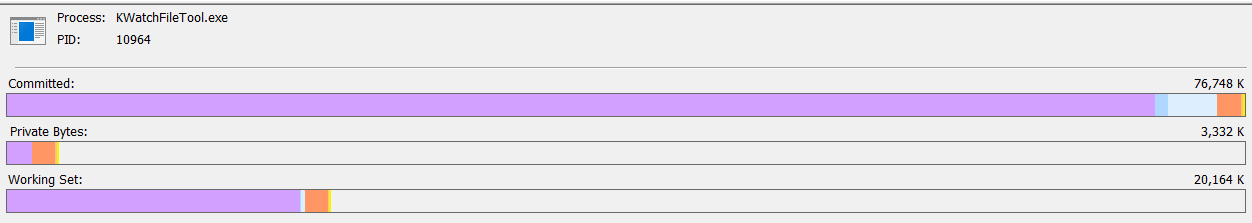
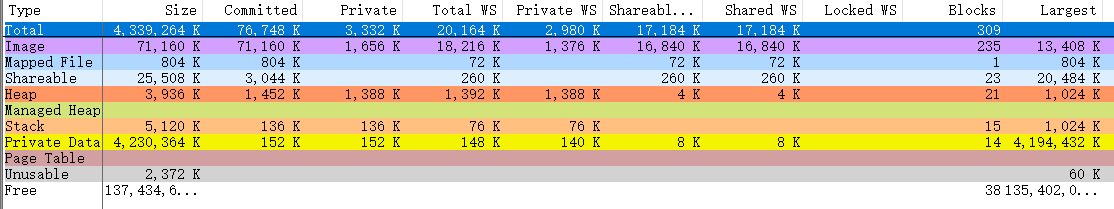


图1.1 VMMap中进程内存总体情况

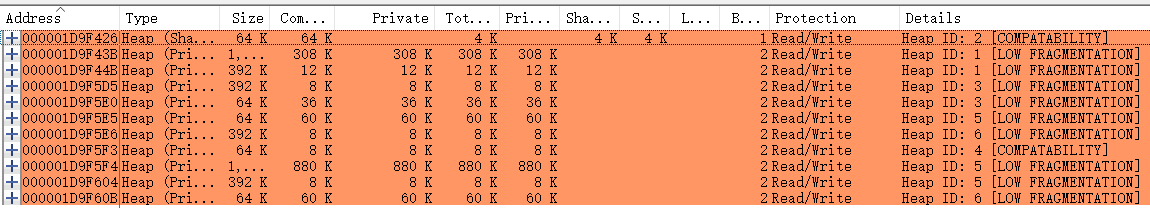
其中顶部为进程基本信息，下部为各类类型空间占用条形图。Committed表示进程向操作系统所申请的需要使用到的虚拟内存大小；Private Bytes表示进程所私用使用的虚拟内存大小；Working Set表示进程目前所使用的物理内存大小。

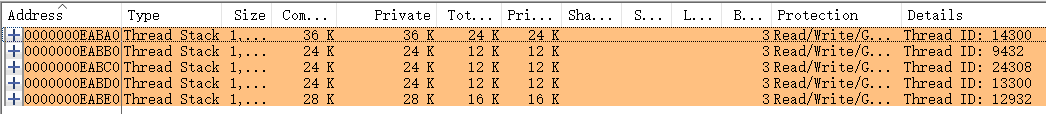
从图中可知，SVG编辑器项目启动后，虚拟内存分配空间大小为76,748K，项目所私有使用的虚拟内存空间大小为3,332K，进程所实际使用的物理内存空间大小为20,164K。

在工具的中部区域，对进程内存总体信息进行了进一步分析



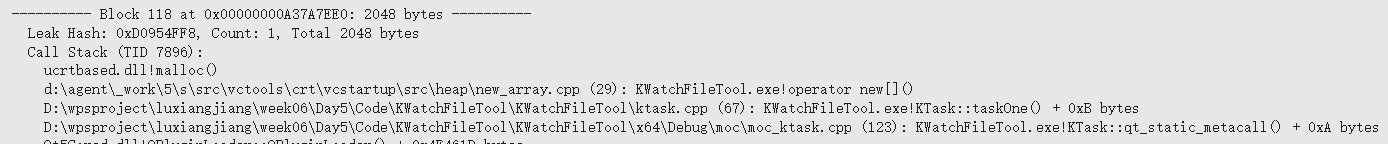
横向表头分别代表总体内存大小、申请内存大小、私有内存大小，总体物理内存带下、私有物理内存大小、共享物理内存大小等类型。纵向表头代表总体、进程自身可执行文件以及包含的所有库、内存映射文件、共享内存、堆区空间、栈区空间等类型。其中new与malloc等分配的空间为堆区域，栈区域空间会主动释放，因此进一步对堆区空间进行分析。





**二、项目内存泄露点**

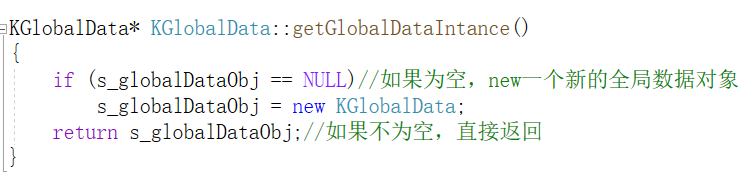
  Visual Leak Detector，它是一款用于 Visual C++免费的内存检测工具。VLD可以得到内存泄漏点的调用堆栈、泄露内存的完整数据，还可以设置内存泄露报告的级别。在VS项目中，加上<vid.h>头文件后运行一次程序，即可得到内存泄露分析结果。可以检测出疑似内存泄露的具体行号，new生成的对象使用完毕后未及时释放，因此发生内存泄露。





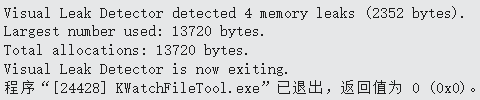
**内容接下一页：**

**◦ 泄漏位置 ：**



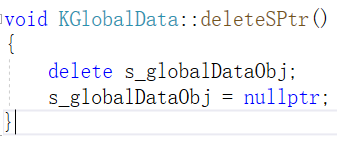
◦ 泄漏原因分析 ：全局静态变量使用后未delete掉

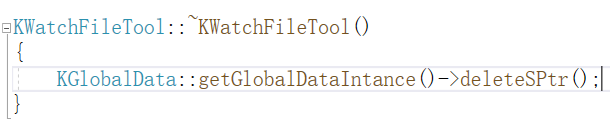
◦ 泄漏解决后：



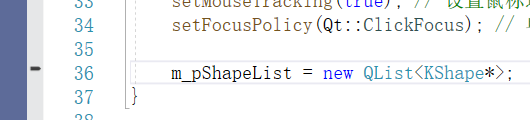
解决方法：在kwatchfiletool.h中添加全局静态变量的析构函数

解决：



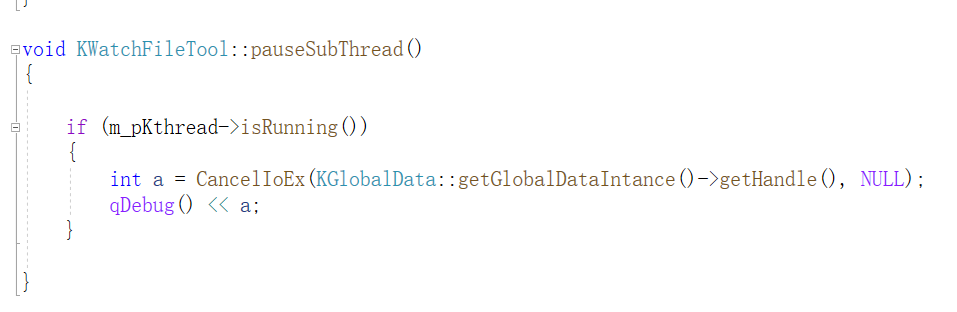


**◦ 泄漏位置 ：**



解决方法：线程中有阻塞循环监听函数，导致线程和KTask对象删除不掉，导致出现内存泄漏

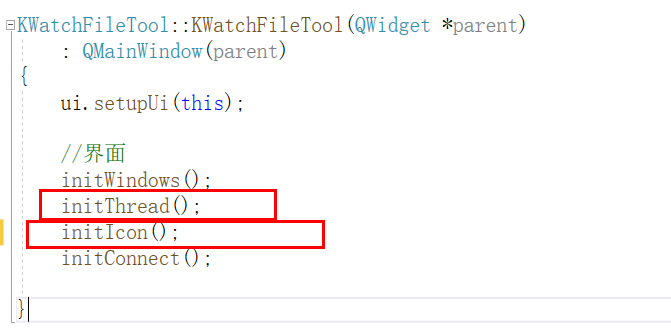
解决：使用函数CancelIoEx（）去掉监听函数的阻塞



然后使用布尔类型允许退出循环，完成delete；



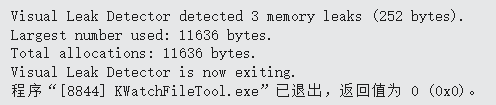
余下的泄露位置：



**泄漏原因分析：**

本人写项目时，不注意细节将子线程和最小化图标的初始化放置在主线程的构造函数中，导致没有时间完成剩余的修改，十分心痛，下次必定谨记教训！！！（鉴于后天的努力，该泄露已修改完成！！！）

修改后：无内存泄漏



**三、UMDH的介绍**

UMDH（user Mode dump heap)是 Debugging Tools for Windows中附带的一个工具。该工具可以分析当前进程在堆上分配的内存，通过命令操作，可以对当前进程分配的每一块内存做日志记录。

**四、VS诊断工具的使用与项目内存分析**

在VS中，可启用自带的诊断工具对项目内存占用情况进行监控与分析，如图4.1。

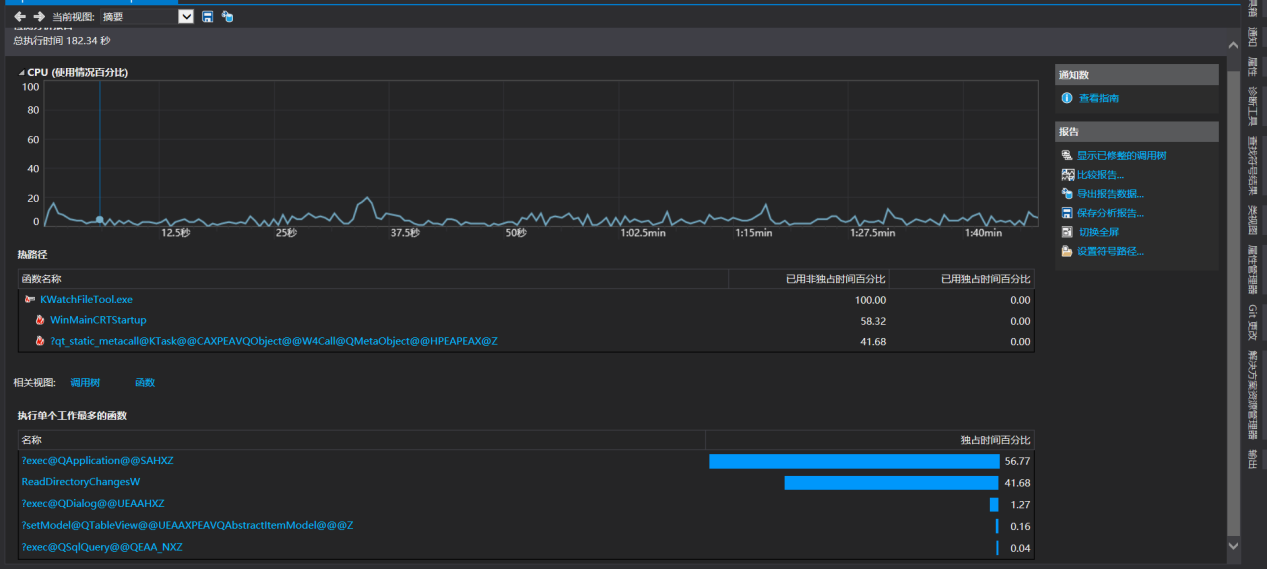


图4.1 VS诊断工具中项目内存分析

还可以通过内存快照，来截取当前时刻的内存使用情况，并对多个时刻的内存使用情况进行对比与分析。

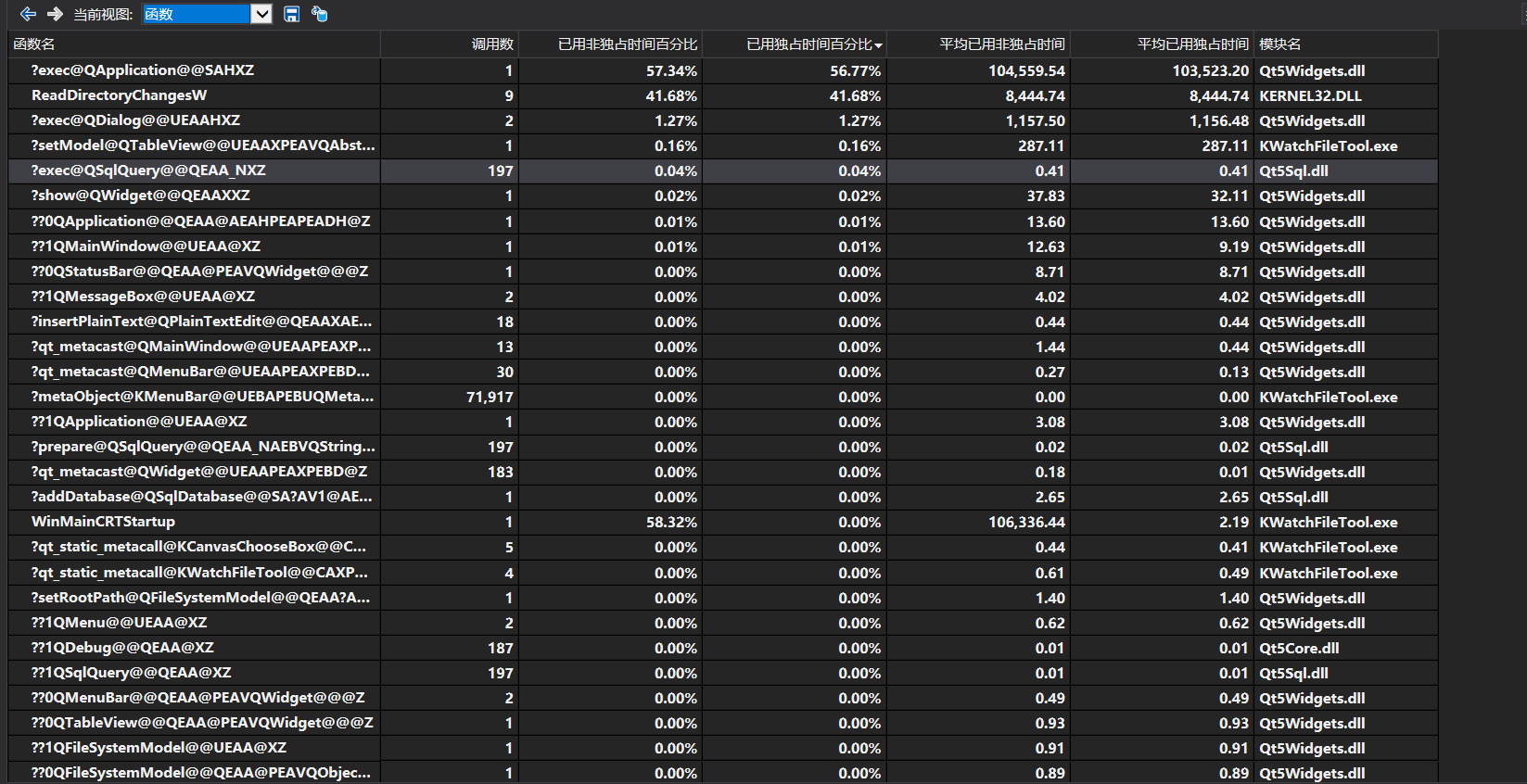


图4.2 VS诊断工具中项目时间占用比

**五、项目内存优化**

我们可以通过优化查找算法、使用数据压缩、使用内存池等方式对内存进行优化。要注意手动创建的对象内存使用完毕后是否释放一些临时变量使用完毕后是否可以及时释放，在使用结构体等复杂的数据类型时要考虑字节对齐、字节序问题，通过调整变量声明顺序来优化内存结构。频繁的new和delete很容易导致内存碎片化，内存管理显得很重要。

尤其是写项目的过程中容易忽略的，需要谨记

1. 全局静态变量在使用后，需要在对应的类中删除对象
2. 多线程的项目中，必须要时刻记得将主线程和子线程的内容分离开来，因为子线程中的对象可能会导致主线程中出现内存泄漏，这是十分严重的错误。
3. 循环阻塞的函数需要一步步解开，第一步解开阻塞，解开阻塞后将继续执行后续的代码，因此在后续代码中存放入退出循环的代码，即可退出循环阻塞函数。
4. new 对象后最好在使用完成后，立刻完成析构。