**VS2017的C++开发心得（十一）调试——内存溢出和内存泄漏（上）**

2018年12月02日 23:36:49 [Mr\_L\_Y](https://me.csdn.net/luoyu510183)

<https://blog.csdn.net/luoyu510183/article/details/84728664>

声明：本文转载blog.csdn.net 中[**Mr\_L\_Y**](https://blog.csdn.net/luoyu510183)的博文，例程略有改动。

VS的调试功能非常强大，一方面是覆盖面很广可以调试本地程序，也可以调试远程的Linux服务器程序。另一方面是内容很丰富，基于汇编调试之上，可以查看内存信息，线程堆栈，甚至GPU、CPU的占用情况，还有DX12的图形调试以及[NVIDIA](https://www.baidu.com/s?wd=NVIDIA&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)的Nsight的CUDA调试。基本满足你从开发到优化到[debug](https://www.baidu.com/s?wd=debug&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)到发布的所有需求。

本章介绍VS下的本地程序的调试方法。

本地程序的调试主要分为两种，一种是主动调试，一种是被动调试。主动调试：即编译并运行，就是“本地Windows调试器”。被动调试：这个需求比较灵活，一般是程序已经发布了release版本，直接双击exe运行而不依赖于VS了。exe运行过程中可能会报错，可能状态异常，这时候我们需要通过VS去链接到这个exe进程进行调试。

下面先介绍如何调试两种最基础的bug：内存溢出和内存泄漏，最后再介绍下如何调试运行中的exe。

内存溢出：也就是内存访问越界，一般出现于数组和指针的使用中。这是一个比较致命的bug。一般情况下，程序都会立即崩溃，提示“0x????????内存读取访问权限冲突”，也可能不会崩溃，但是你的程序总是会有时候运行异常。

内存泄漏：相较于内存溢出，它的危害要小一些。泄漏就是申请的内存没有合理回收，导致出现不可使用的内存片段，大量的泄漏最终还是会导致程序无法申请到新的内存而崩溃。

下面一段代码则一下同时出现这两种错误：

// DebugMemory.cpp

#include <iostream>

//For Debug

#define CRTDBG\_MAP\_ALLOC

#include <stdlib.h>

#include <crtdbg.h>

#ifdef \_DEBUG

#define DBG\_NEW new ( \_NORMAL\_BLOCK , \_\_FILE\_\_ , \_\_LINE\_\_ )

// Replace \_NORMAL\_BLOCK with \_CLIENT\_BLOCK if you want the

// allocations to be of \_CLIENT\_BLOCK type

#else

#define DBG\_NEW new

#endif

//Notice: test() Error!!! It is an incorrect sample for debug demo.

double test()

{

double \*t = new double[2]; //Error!!! New memory,But memory is not freed.

return t[2]; //Error!!! Access cross-border, memory overflow.

}

//main

int main()

{

std::cout << "Debug Demo: Detected memory leaks!\n";

double k = test();

// For debug

\_CrtDumpMemoryLeaks(); // Detected memory leaks!

}

在Output 视图中报内存泄漏：

Detected memory leaks!

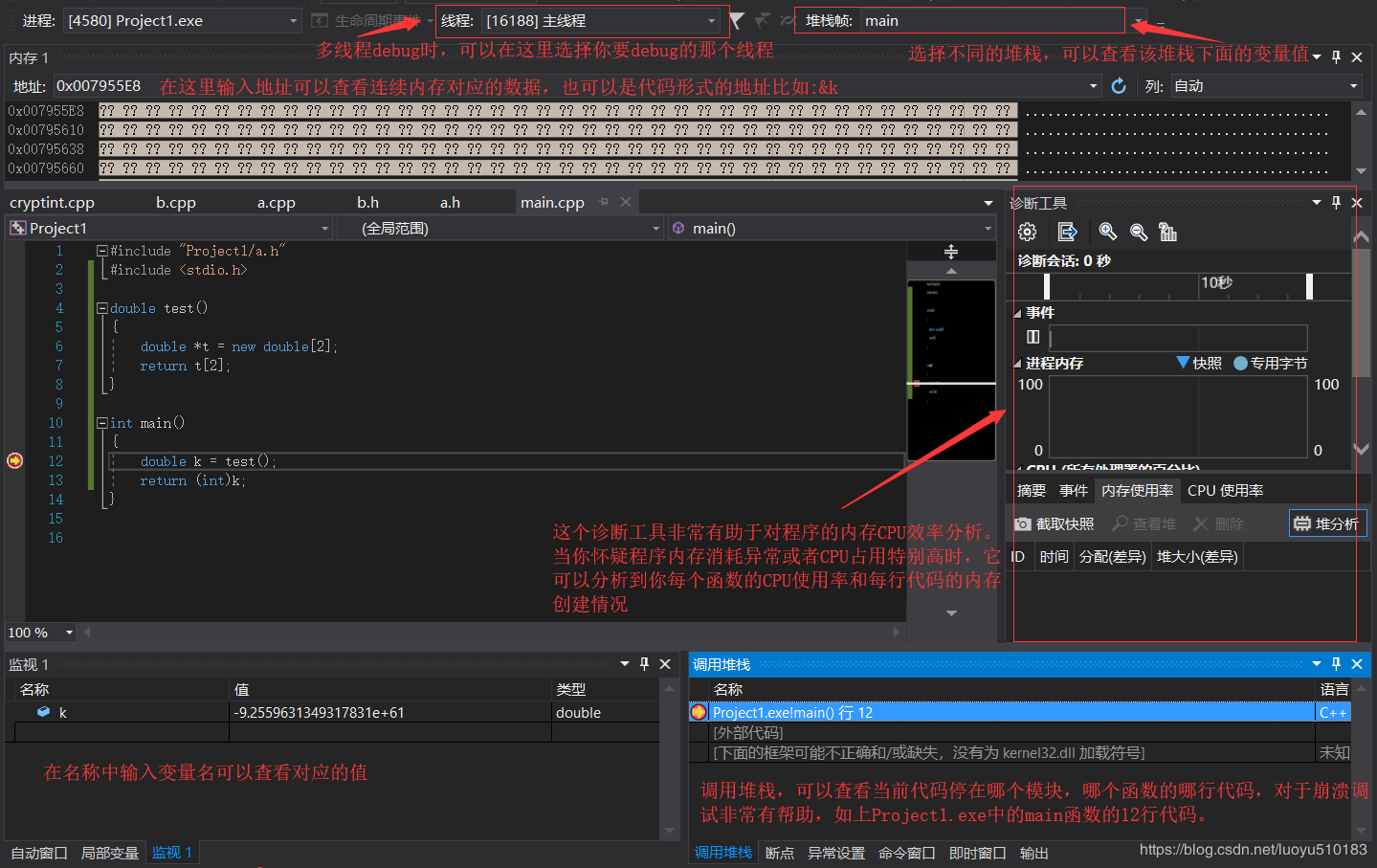
Dumping objects ->

{153} normal block at 0x00B9CAB0, 16 bytes long.

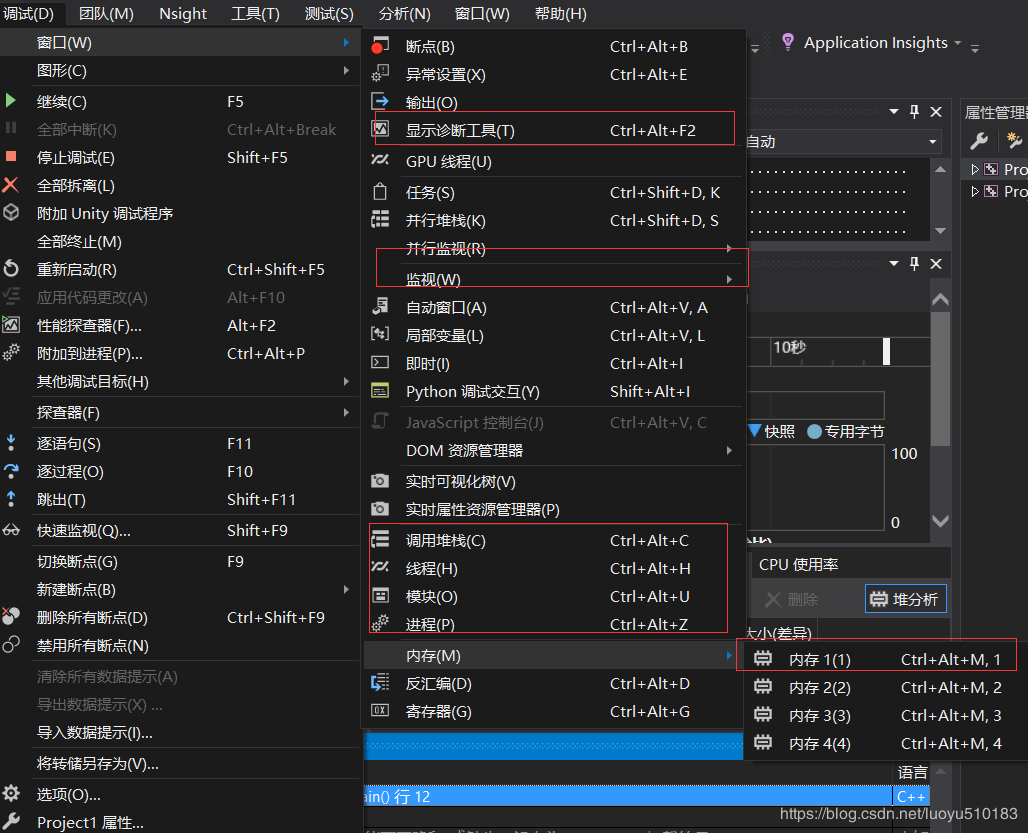
Data: < > CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD

Object dump complete.

在开始debug之前，先看看我的VSdebug[界面](https://www.baidu.com/s?wd=%E7%95%8C%E9%9D%A2&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)下有哪些工具栏帮助我们debug代码错误：



上图中的窗口和工具栏可以在调试菜单下打开：



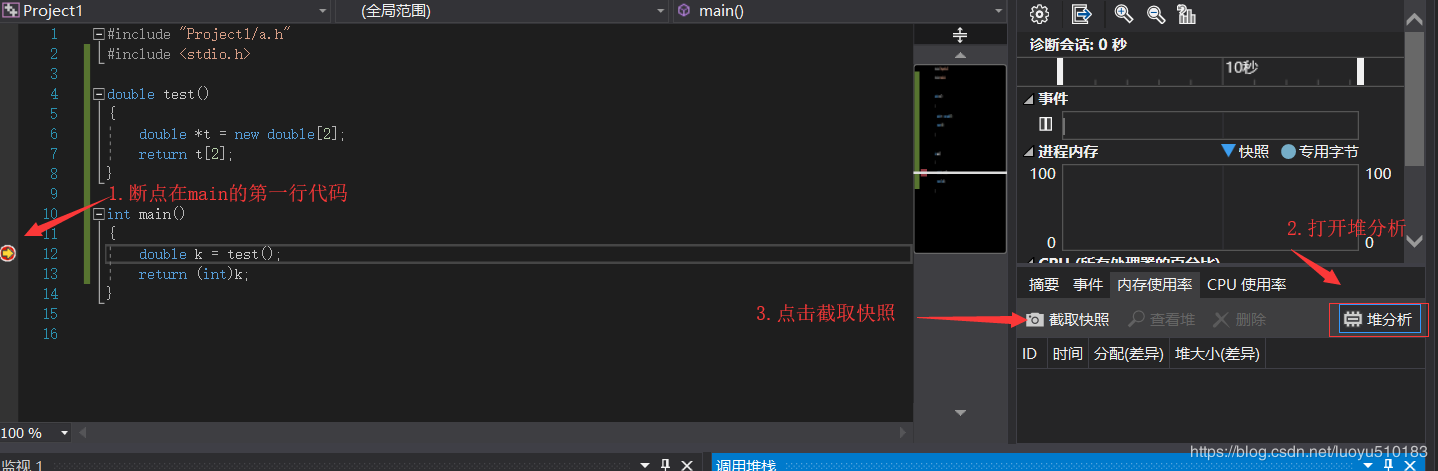
准备就绪，但先来解决内存泄漏的问题，由于新版VS的帮助这个bug现在特别容易解决。

上面的代码中，在test()函数中new了2个double的内存空间，但没有delete这个内存。对于我们来讲，这块内存的指针存在于test函数栈上，当函数运行结束后指针就释放掉了，我们外面的程序已经无法使用这块内存。对于系统来讲，这块内存已经分配给了我们的程序，但并没有被程序释放掉，所以也处于不可用的状态。这样一来这2xdouble的空间出于了完全不可用的状态，经常这样申请内存会导致我们能用的内存越来越少。

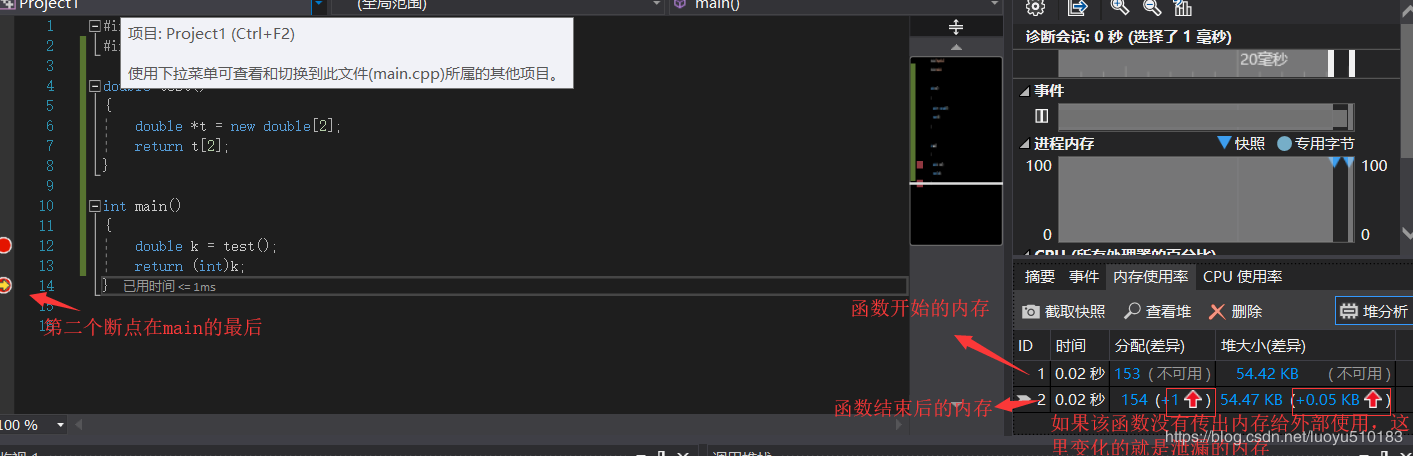
通过代码分析当然可以找出这个泄漏，但太慢了。上面的诊断工具中的“内存使用率”的“堆分析”可以迅速帮我们定位堆上没有合理释放的内存。

如下：

先在函数开始时截取当前的堆快照：

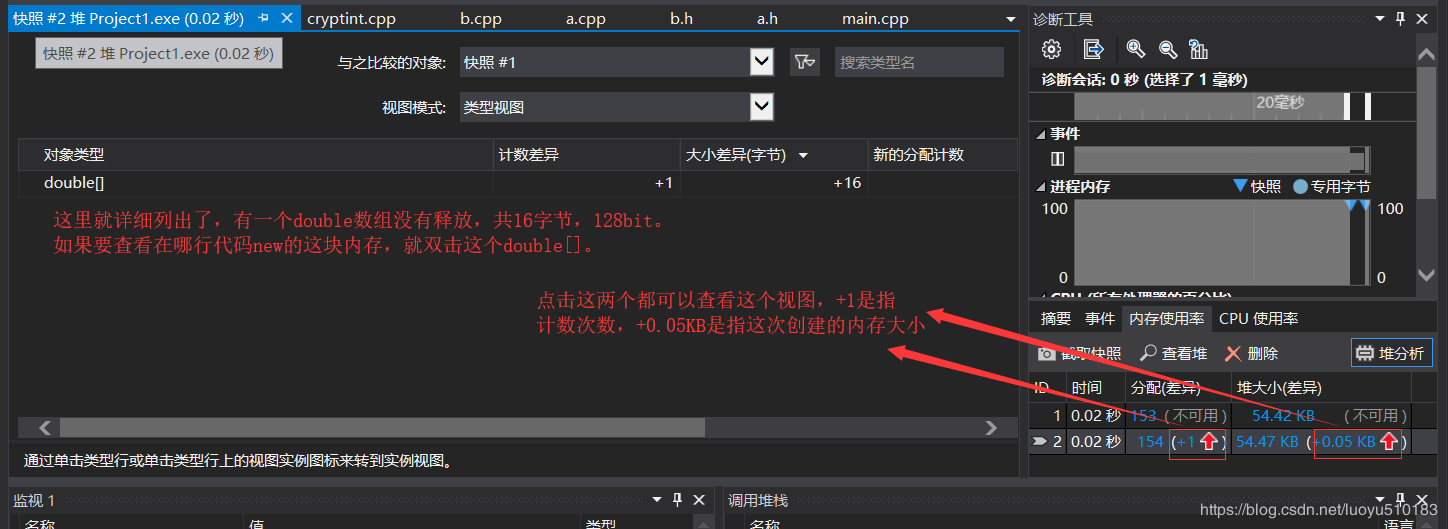


然后在函数结束时，再截取一次堆的快照：

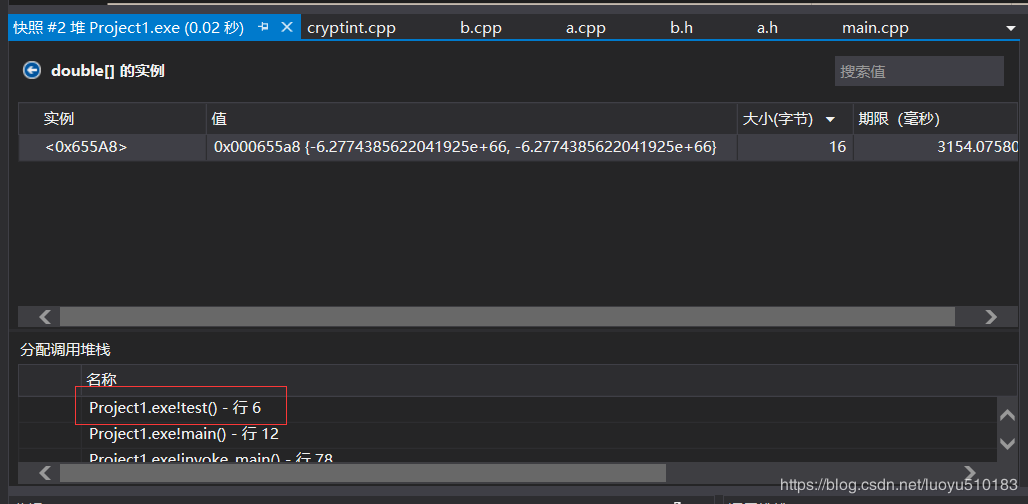


好，因为main就是主函数，运行完main以后应该释放掉所有的申请的内存，但上图中显示有0.05kb的内存还没有被释放掉。

接下来看看怎么定位内存泄漏的代码。先查看堆视图：



然后在堆栈中可以看到这块内存是在main的第6行代码中申请的：



双击上图中的方框内容，VS就会定位到这个堆栈对应的代码也就是：

double \*t = new double[2];

知道了泄漏的代码以后，就可以根据你的需求在合适的地方把这块内存给delete掉。

补充一点，过去有人告诉你这样new的数组，在删除的时候要用delete[]来删除。现在不需要了，直接delete t就可以了。

这篇就到这里，下一篇再介绍怎么debug内存溢出的情况。