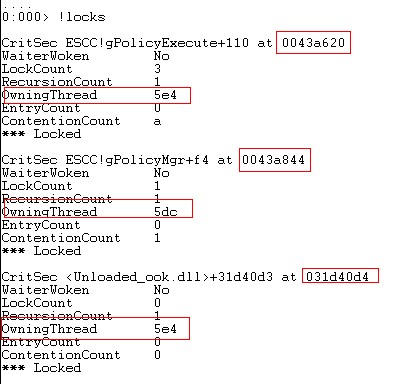
死锁:是指两个或两个以上的线程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。



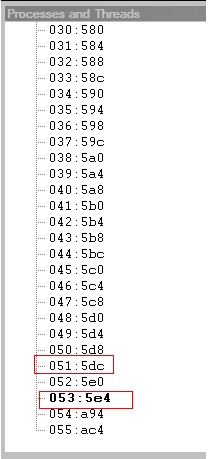
1）  先用**!locks**查看所有的线程占用的锁



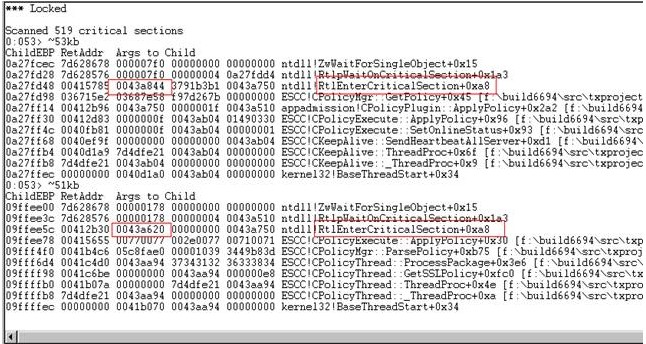
这里可以看到有三个线程正在等待三个锁，第一个线程等待的锁是0043a620，但被5e4这条线程占用，第二个线程等待的锁是0043a844，但被5dc线程占用，

第三个线程等待的锁是031d40d4，也被5dc线程占用。

2）接着，我们需要查看5e4线程和5dc线程的id，具体可以通过查看工具栏中Processes and Threads，如下图所示，5e4的线程的id为53,5dc线程的id为51。

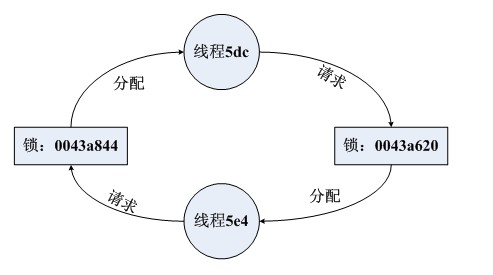


3）分别输入~53kb和~51kb查看这个两个线程的调用栈，结果如下图所示



由数据可知，5e4线程正在等待一把0043a844的锁，而5dc线程也正在等待锁0043a620。

4）结合第一步获取的信息可知，**5e4线程要去获取已经被5dc占用的锁0043a844，而5dc又要去获取已经被5e4占用的锁0043a620**，如此形成环路，就产生了死锁



当！locks出现问的时候

情况一：

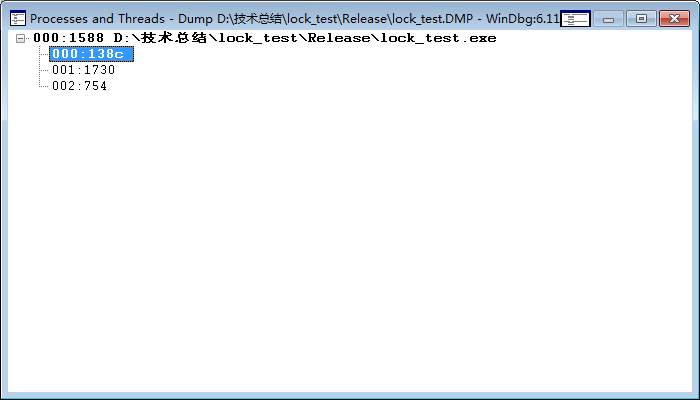
ChildEBP RetAddr  Args to Child               
065d5730 7c92df5a 7c939b23 000008e0 00000000 ntdll!KiFastSystemCallRet (FPO: [0,0,0])  
065d5734 7c939b23 000008e0 00000000 00000000 ntdll!NtWaitForSingleObject+0xc (FPO: [3,0,0])  
065d57bc 7c921046 0079f570 7d5bd275 7d79f570 ntdll!RtlpWaitForCriticalSection+0x132 (FPO: [1,26,4])  
065d57c4 7d5bd275 7d79f570 05cc7bc5 065d5814 ntdll!RtlEnterCriticalSection+0x46 (FPO: [1,0,0])  
WARNING: Stack unwind information not available. Following frames may be wrong.  
065d57d8 7d5bd238 7d5980f4 05cc7bc5 05ccf050 SHELL32!ILFindLastID+0x409  
065d57f4 7d5bd833 00000000 00000000 05cc7bc5 SHELL32!ILFindLastID+0x3cc  
065d5a40 7d5bd8de 00000000 05cc7c69 065d5a7c SHELL32!ILFindLastID+0x9c7  
065d5c98 7d5bdb02 00000000 05cc7bc5 7d597034 SHELL32!ILFindLastID+0xa72  
065d5cc0 7d5bd5e2 00000000 05cc7bc5 00000000 SHELL32!ILFindLastID+0xc96  
065d5ce8 7d5bd7c2 05cbc240 05cc8340 00000000 SHELL32!ILFindLastID+0x776  
065d5d0c 7d5bb846 05cc0800 05cc7bac 00000000 SHELL32!ILFindLastID+0x956  
065d5d38 7d5bb89f 05cc7b98 00000000 7d597034 SHELL32!ILCombine+0x241  
065d5d5c 7d5bbd12 04506770 05cc7b98 00000000 SHELL32!ILCombine+0x29a  
065d5d80 7d5bc6f7 04506770 05cc7b98 00000000 SHELL32!SHGetDesktopFolder+0xba  
065d5da4 7d5bc73c 00000000 05cc72e0 7d597034 SHELL32!ILRemoveLastID+0x50  
065d5dc0 7d5c10ac 05cc72e0 7d597034 065d5df0 SHELL32!ILRemoveLastID+0x95  
065d6010 7d5c1052 05cc72e0 065d79bc 00000002 SHELL32!SHGetPathFromIDListW+0x6c  
065d6024 04f56f03 05cc72e0 065d79bc c54b6ba4 SHELL32!SHGetPathFromIDListW+0x12

分析如下：

0:009> !locks  
NTSDEXTS: Unable to resolve ntdll!RTL\_CRITICAL\_SECTION\_DEBUG type  
NTSDEXTS: Please check your symbols  
0:009> dt RTL\_CRITICAL\_SECTION 7d79f570  
XXXXX!RTL\_CRITICAL\_SECTION  
   +0x000 DebugInfo      : 0x00153ab8 \_RTL\_CRITICAL\_SECTION\_DEBUG  
   +0x004 LockCount      : 1  
   +0x008 RecursionCount   : 1  
   +0x00c OwningThread     : 0x00000964  
   +0x010 LockSemaphore    : 0x000008e0  
   +0x014 SpinCount      : 0

**看来是0x964线程持有锁，对应查看具体的线程**

情况二：



1 Id: 1588.1730 Suspend: 0 Teb: 7ffdd000 Unfrozen

ChildEBP RetAddr Args to Child

WARNING: Stack unwind information not available. Following frames may be wrong.

009ffcd4 7759215c 00000000 00000000 00000000 ntdll!KiFastSystemCallRet

009ffcfc 012d101d 012d3370 00000000 76d3ee1c ntdll!EtwEventEnabled+0xd9

009ffd08 76d3ee1c 00000000 009ffd54 775c37eb lock\_test!thread1+0x1d (FPO: [1,0,1]) (CONV: stdcall) [d:\技术总结\lock\_test\lock\_test\lock\_test.cpp @ 16]

009ffd14 775c37eb 00000000 76a98121 00000000 kernel32!BaseThreadInitThunk+0x12

009ffd54 775c37be 012d1000 00000000 00000000 ntdll!RtlInitializeExceptionChain+0xef

009ffd6c 00000000 012d1000 00000000 00000000 ntdll!RtlInitializeExceptionChain+0xc2

2 Id: 1588.754 Suspend: 0 Teb: 7ffdc000 Unfrozen

ChildEBP RetAddr Args to Child

WARNING: Stack unwind information not available. Following frames may be wrong.

0073fed4 7759215c 00000000 00000000 00000000 ntdll!KiFastSystemCallRet

0073fefc 012d104d 012d3388 00000000 76d3ee1c ntdll!EtwEventEnabled+0xd9

0073ff08 76d3ee1c 00000000 0073ff54 775c37eb lock\_test!thread2+0x1d (FPO: [1,0,1]) (CONV: stdcall) [d:\技术总结\lock\_test\lock\_test\lock\_test.cpp @ 25]

0073ff14 775c37eb 00000000 76458321 00000000 kernel32!BaseThreadInitThunk+0x12

0073ff54 775c37be 012d1030 00000000 00000000 ntdll!RtlInitializeExceptionChain+0xef

0073ff6c 00000000 012d1030 00000000 00000000 ntdll!RtlInitializeExceptionChain+0xc2

0:000> uf 012d3388 ：显示函数的反汇编代码

Flow analysis was incomplete, some code may be missing

lock\_test!cs1:

012d3388 a04f3500fa mov al,byte ptr ds:[FA00354Fh]

0:000> dt RTL\_CRITICAL\_SECTION 012d3388

lock\_test!RTL\_CRITICAL\_SECTION

+0x000 DebugInfo : 0x00354fa0 \_RTL\_CRITICAL\_SECTION\_DEBUG

+0x004 LockCount : -6

+0x008 RecursionCount : 1

+0x00c OwningThread : 0x00001730

+0x010 LockSemaphore : 0x0000003c

+0x014 SpinCount : 0

2）

0:000> uf 012d3370 显示函数的反汇编代码

Flow analysis was incomplete, some code may be missing

lock\_test!cs2:

012d3370 c84f3500 enter 354Fh,0

012d3374 fa cli

0:000> dt RTL\_CRITICAL\_SECTION 012d3370

lock\_test!RTL\_CRITICAL\_SECTION

+0x000 DebugInfo : 0x00354fc8 \_RTL\_CRITICAL\_SECTION\_DEBUG

+0x004 LockCount : -6

+0x008 RecursionCount : 1

+0x00c OwningThread : 0x00000754

+0x010 LockSemaphore : 0x00000040

+0x014 SpinCount : 0