WinDbg 内核调试常用命令(1)

本章介绍内核调试下的常用命令，内容稍多，准备分为两篇来介绍。第一篇主要涉及查看目标机状态、进程相关、线程相关命令。介绍每个命令的主要作用，以及常用方式，不会涉及详细的命令参数，目的是能快速上手熟悉内核调试下的常用操作，而不是替代帮助文件。

查看目标机

调试过程中经常需要查看目标机的状态，比如目标系统类型等。下面几个是内核调试时常用的命令，可以方面的查看当前目标机状态，其它如查看进程列表、查看内核模块等命令在接下来的几节中在详细介绍。

vertarget

vertarget 命令可以显示目标系统的基本信息，如系统版本、计算机名、内核基址等。

```
kd> vertarget
Windows Server 2003 Kernel Version 3790 (Service Pack 1) UP Free x86 compatible
Built by: 3790.srv03_sp1_rtm.050324-1447
Machine Name:
Kernel base = 0x80800000 PsLoadedModuleList = 0x808a8e48
Debug session time: Sat Dec 20 17:16:07.913 2008 (GMT+8)
System Uptime: 0 days 0:35:04.726
```

从输出来看，目标机为 2003SP1 系统，非 Checked 版，内核基址为 0x80800000，计算机名没有获取到。

dg

dg 命令主要显示选择子的详细信息。

```
kd> r
eax=00000001 ebx=00034e04 ecx=8089fc8c edx=000003f8 esi=00000005 edi=0cc843f0
eip=8081d97e esp=8089d580 ebp=8089d590 iopl=0         nv up ei pl nz na po nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000202
nt!RtlpBreakWithStatusInstruction:
8081d97e int     3
kd> dg @cs
                              P Si Gr Pr Lo
Sel    Base     Limit     Type    l ze an es ng Flags
---- -------- -------- ---------- - -- -- -- -- --------
0008 00000000 ffffffff Code RE Ac 0 Bg Pg P  Nl 00000c9b
kd> dg @ds
                              P Si Gr Pr Lo
Sel    Base     Limit     Type    l ze an es ng Flags
---- -------- -------- ---------- - -- -- -- -- --------
0023 00000000 ffffffff Data RW Ac 3 Bg Pg P  Nl 00000cf3
```

```
kd> dg @fs
                            P Si Gr Pr Lo
Sel    Base     Limit     Type    l ze an es ng Flags
---- -------- -------- ---------- - -- -- -- -- --------
0030 ffdff000 00001fff Data RW Ac 0 Bg Pg P  Nl 00000c93
```

从 dg 命令的输出信息可以看到 cs 段和 ds 段范围都是 0x00000000-0xffffffff，大小为 4G。

!cpuinfo

!cpuinfo 命令显示 CPU 信息。

```
kd> !cpuinfo
CP  F/M/S Manufacturer  MHz PRCB Signature    MSR 8B Signature Features
 0  6,15,11 GenuineIntel 2319 7fffffff00000000  0000000000000000 80013bbf
                  Cached Update Signature 0000000000000000
                 Initial Update Signature 7fffffff00000000
```

!pcr 和!prcb

!pcr 命令显示处理器控制域（Processor Control Region）信息，也就是 KPCR 结构信息，每个 CPU 对应一个 KPCR 结构，可以在命令中指定需要显示的 CPU 序号，不指定则显示当前 CPU 的 PCR 信息。

!prcb 命令显示处理器控制块（Processor Control Block）信息，也就是 KPRCB 结构信息，同样，每个 CPU 对应一个 KPRCB 结构，该结构紧接在 KPCR 结构后面，由 KPCB 结构中的 PrcbData 表示。命令中可以指定 CPU 序号。

```
lkd> !pcr
KPCR for Processor 0 at 81ef8800:
   Major 1 Minor 1
    NtTib.ExceptionList: 9f840a3c
        NtTib.StackBase: 00000000
       NtTib.StackLimit: 00000000
     NtTib.SubSystemTib: 80154000
          NtTib.Version: 00f54ca0
      NtTib.UserPointer: 00000001
          NtTib.SelfTib: 7ffdd000

                SelfPcr: 81ef8800
                   Prcb: 81ef8920
                   Irql: 00000002
                    IRR: 00000000
                    IDR: ffffffff
          InterruptMode: 00000000
                    IDT: 83112400
                    GDT: 83112000
                    TSS: 80154000
```

```
          CurrentThread: 8497e030
             NextThread: 00000000
             IdleThread: 81efc640

              DpcQueue:
lkd> !pcr 1
KPCR for Processor 1 at 805d1000:
    Major 1 Minor 1
     NtTib.ExceptionList: 9f840a3c
         NtTib.StackBase: 00000000
        NtTib.StackLimit: 00000000
      NtTib.SubSystemTib: 805d3130
          NtTib.Version: 00cfa74e
       NtTib.UserPointer: 00000002
          NtTib.SelfTib: 7ffdd000

                SelfPcr: 805d1000
                   Prcb: 805d1120
                   Irql: 00000002
                    IRR: 00000000
                    IDR: ffffffff
          InterruptMode: 00000000
                    IDT: 805d89d0
                    GDT: 805d85d0
                    TSS: 805d3130

          CurrentThread: 84917030
             NextThread: 00000000
             IdleThread: 805d51e0

              DpcQueue:
```

输出包括内核 SEH 链表头、TSS、IDT 等重要信息。有了 PCR 地址，还可以通过 dt 命令直接显示 KPCR 结构，或者直接用?? @$pcr 命令显示当前 PCR 信息。

```
lkd> dt nt!_KPCR 805d1000
   +0x000 NtTib          : _NT_TIB
   +0x000 Used_ExceptionList : 0x9f840a3c _EXCEPTION_REGISTRATION_RECORD
   +0x004 Used_StackBase   : (null)
   +0x008 Spare2         : (null)
   +0x00c TssCopy        : 0x805d3130
   +0x010 ContextSwitches  : 0xda0ead
   +0x014 SetMemberCopy    : 2
   +0x018 Used_Self       : 0x7ffdd000
   +0x01c SelfPcr         : 0x805d1000 _KPCR
   +0x020 Prcb           : 0x805d1120 _KPRCB
   +0x024 Irql           : 0x2 ''
   +0x028 IRR            : 0
   +0x02c IrrActive       : 0
```

```
   +0x030 IDR            : 0xffffffff
   +0x034 KdVersionBlock   : (null)
   +0x038 IDT            : 0x805d89d0 _KIDTENTRY
   +0x03c GDT            : 0x805d85d0 _KGDTENTRY
   +0x040 TSS            : 0x805d3130 _KTSS
   +0x044 MajorVersion     : 1
   +0x046 MinorVersion     : 1
   +0x048 SetMember       : 2
   +0x04c StallScaleFactor : 0x91d
   +0x050 SpareUnused     : 0 ''
   +0x051 Number          : 0x1 ''
   +0x052 Spare0          : 0 ''
   +0x053 SecondLevelCacheAssociativity : 0 ''
   +0x054 VdmAlert        : 0
   +0x058 KernelReserved   : [14] 0
   +0x090 SecondLevelCacheSize : 0
   +0x094 HalReserved     : [16] 1
   +0x0d4 InterruptMode    : 0
   +0x0d8 Spare1          : 0 ''
   +0x0dc KernelReserved2  : [17] 0
   +0x120 PrcbData        : _KPRCB
lkd> ?? @$pcr
struct _KPCR * 0x81ef8800
   +0x000 NtTib          : _NT_TIB
   +0x000 Used_ExceptionList : 0x9f840a3c _EXCEPTION_REGISTRATION_RECORD
   +0x004 Used_StackBase   : (null)
   +0x008 Spare2          : (null)
   +0x00c TssCopy         : 0x80154000
   +0x010 ContextSwitches  : 0x121ea84
   +0x014 SetMemberCopy    : 1
   +0x018 Used_Self       : 0x7ffdd000
   +0x01c SelfPcr         : 0x81ef8800 _KPCR
   +0x020 Prcb            : 0x81ef8920 _KPRCB
   +0x024 Irql            : 0x2 ''
   +0x028 IRR             : 0
   +0x02c IrrActive       : 0
   +0x030 IDR             : 0xffffffff
   +0x034 KdVersionBlock   : 0x81ef7c68
   +0x038 IDT             : 0x83112400 _KIDTENTRY
   +0x03c GDT             : 0x83112000 _KGDTENTRY
   +0x040 TSS             : 0x80154000 _KTSS
   +0x044 MajorVersion     : 1
   +0x046 MinorVersion     : 1
   +0x048 SetMember       : 1
   +0x04c StallScaleFactor : 0x91d
   +0x050 SpareUnused     : 0 ''
   +0x051 Number          : 0 ''
   +0x052 Spare0          : 0 ''
```

```
   +0x053 SecondLevelCacheAssociativity : 0 ''
   +0x054 VdmAlert        : 0
   +0x058 KernelReserved   : [14] 0
   +0x090 SecondLevelCacheSize : 0
   +0x094 HalReserved     : [16] 0
   +0x0d4 InterruptMode    : 0
   +0x0d8 Spare1          : 0 ''
   +0x0dc KernelReserved2  : [17] 0
   +0x120 PrcbData        : _KPRCB
```

!prcb 命令显示 KPRCB 结构，里面包含了当前线程、Idle 线程等重要信息。同样可以用 dt 命令或?? @$prcb 命令显示 KPRCB 结构的详细信息。

```
lkd> !prcb
PRCB for Processor 0 at 81ef8920:
Current IRQL -- 0
Threads--  Current 8497e030 Next 00000000 Idle 81efc640
Number 0 SetMember 1
Interrupt Count -- 00724738
Times -- Dpc    00000875 Interrupt 00000641
        Kernel 0007b00b User     00005e35
```

所有和特定 CPU 相关的信息基本都保存在 KPCR 和 KPRCB 中，比如 DPC。

!idt

!idt 命令显示中断服务表，可以指定中断号显示，也可以显示全部的中断服务表。

```
kd> !idt

Dumping IDT:

30: 80a72eb8 hal!HalpClockInterrupt
31: 81e28044 i8042prt!I8042KeyboardInterruptService (KINTERRUPT 81e28008)
33: 81dc0164 serial!SerialCIsrSw (KINTERRUPT 81dc0128)
38: 80a6d7bc hal!HalpProfileInterrupt
39: 81fb7bf4 ACPI!ACPIInterruptServiceRoutine (KINTERRUPT 81fb7bb8)
3a: 81eea9f4 vmsrvc+0x19B8 (KINTERRUPT 81eea9b8)
3b: 81e53044 NDIS!ndisMIsr (KINTERRUPT 81e53008)
3c: 81e28dd4 i8042prt!I8042MouseInterruptService (KINTERRUPT 81e28d98)
3e: 81f63044 atapi!IdePortInterrupt (KINTERRUPT 81f63008)
3f: 81f632cc atapi!IdePortInterrupt (KINTERRUPT 81f63290)

kd> !idt 3

Dumping IDT:

03: 8082446a nt!KiTrap03
```

```
kd> !idt -a

Dumping IDT:

00: 80823ede nt!KiTrap00
01: 8082405a nt!KiTrap01
02: Task Selector = 0x0058
03: 8082446a nt!KiTrap03
......
2d: 80824346 nt!KiDebugService
2e: 808231ba nt!KiSystemService
2f: 80826574 nt!KiTrap0F
30: 80a72eb8 hal!HalpClockInterrupt
31: 81e28044 i8042prt!I8042KeyboardInterruptService (KINTERRUPT 81e28008)
......
ff: 80823060 nt!KiUnexpectedInterrupt207
```

!irql

!irql 命令显示目标系统中断到 WinDbg 时的中断请求级（IRQL），在 Windows2003 及以后的系统上可用，调试和 IRQL 相关的程序时可能会用到。

```
kd> !irql
Debugger saved IRQL for processor 0x0 -- 28 (CLOCK2_LEVEL) [Interval clock 2 level]
```

!running

!running 命令显示所有 CPU 上正在运行的线程信息，便于了解系统当前的情况，如果是分析蓝屏文件，则可以看到蓝屏时系统正在执行什么线程。

```
kd> !running -it

System Processors 1 (affinity mask)
  Idle Processors 1

All processors idle.

        Prcbs     Current   Next
  0    ffdff120  8089fd80          ..............

ChildEBP RetAddr
8089d57c 8082050c nt!RtlpBreakWithStatusInstruction
8089d57c 80a7253e nt!KeUpdateSystemTime+0x129
8089d600 80820a45 hal!HalProcessorIdle+0x2
8089d604 00000000 nt!KiIdleLoop+0xa
```

从输出可以看出是 WinDbg 主动中断目标系统的。

!gflag

!gflag 命令显示、设置系统全局标志，用于控制系统行为，和 WinDbg 自带的 gflags.exe 工具类似，不过设置后只对当前调试会话有效。

```
kd> !gflag
Current NtGlobalFlag contents: 0x00000000
kd> !gflag -?
usage: !gflag [-? | flags]
Flags may either be a single hex number that specifies all
32-bits of the GlobalFlags value, or it can be one or more
arguments, each beginning with a + or -, where the + means
to set the corresponding bit(s) in the GlobalFlags and a -
means to clear the corresponding bit(s).  After the + or -
may be either a hex number or a three letter abbreviation
for a GlobalFlag.  Valid abbreviations are:
    soe - Stop On Exception
    sls - Show Loader Snaps
    dic - Debug Initial Command
    shg - Stop on Hung GUI
    htc - Enable heap tail checking
    hfc - Enable heap free checking
    hpc - Enable heap parameter checking
    hvc - Enable heap validation on call
    vrf - Enable application verifier
    ptg - Enable pool tagging
    htg - Enable heap tagging
    ust - Create user mode stack trace database
    kst - Create kernel mode stack trace database
    otl - Maintain a list of objects for each type
    htd - Enable heap tagging by DLL
    dse - Disable stack extensions
    d32 - Enable debugging of Win32 Subsystem
    ksl - Enable loading of kernel debugger symbols
    dps - Disable paging of kernel stacks
    scb - Enable system critical breaks
    dhc - Disable Heap Coalesce on Free
    ece - Enable close exception
    eel - Enable exception logging
    eot - Enable object handle type tagging
    hpa - Place heap allocations at ends of pages
    dwl - Debug WINLOGON
    ddp - Disable kernel mode DbgPrint output
    cse - Early critical section event creation
    ltd - Load DLLs top-down
    bhd - Enable bad handles detection
    dpd - Disable protected DLL verification
kd> !gflag +soe
```

```
New NtGlobalFlag contents: 0x00000001
    soe - Stop On Exception
kd> !gflag
Current NtGlobalFlag contents: 0x00000001
    soe - Stop On Exception
```

进程相关

内核调试时，经常会和进程打交道，比如查看进程列表、查找进程 EPROCESS、切换进程空间等。

!process

!process 命令显示进程信息，是个常用命令。可以显示 EPROCESS、进程 ID、句柄表、页目录、线程列表等重要信息。有了 EPROCESS 地址，也能用 dt nt!_EPROCESS xxxxxxxx 命令显示 EPROCESS 结构的详细内容，?? @$proc 可以显示当前进程的 EPROCESS 结构。

!process 0 0 显示进程列表，只显示每个进程的基本信息。

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS 81f9b9e8  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 00039000  ObjectTable: e1000d50  HandleCount: 203.
    Image: System

PROCESS 81daa6d0  SessionId: none  Cid: 0148    Peb: 7ffde000  ParentCid: 0004
    DirBase: 1cf1c000  ObjectTable: e127d3e0  HandleCount:  18.
    Image: SMSS.EXE

PROCESS 81da61d0  SessionId: 0  Cid: 0180    Peb: 7ffd5000  ParentCid: 0148
    DirBase: 1e26b000  ObjectTable: e127d7c0  HandleCount: 330.
    Image: csrss.exe

......

PROCESS 8183c408  SessionId: 0  Cid: 0740    Peb: 7ffd9000  ParentCid: 027c
    DirBase: 0528f000  ObjectTable: e174b250  HandleCount: 156.
    Image: WMIPRVSE.EXE

PROCESS 81f30d88  SessionId: 0  Cid: 0320    Peb: 7ffde000  ParentCid: 0518
    DirBase: 09ce0000  ObjectTable: e10d3ac0  HandleCount:  32.
    Image: conime.exe
```

!process xxxxxxxx 显示指定进程的所有信息。!process xxxxxxxx 0 则显示指定进程的基本信息。xxxxxxxx 可以为 EPROCESS，也可以为进程 ID。

```
kd> !process 8182aa20
PROCESS 8182aa20  SessionId: 0  Cid: 0084    Peb: 7ffde000  ParentCid: 0668
    DirBase: 0ac9c000  ObjectTable: e1264a88  HandleCount:  41.
    Image: calc.exe
```

```
    VadRoot 818473e0 Vads 52 Clone 0 Private 155. Modified 7. Locked 0.
    DeviceMap e17a1a50
    Token                             e1a78528
    ElapsedTime                       00:00:03.515
    UserTime                          00:00:00.000
    KernelTime                        00:00:00.000
    QuotaPoolUsage[PagedPool]         30260
    QuotaPoolUsage[NonPagedPool]      2080
    Working Set Sizes (now,min,max)  (873, 50, 345) (3492KB, 200KB, 1380KB)
    PeakWorkingSetSize                874
    VirtualSize                       28 Mb
    PeakVirtualSize                   30 Mb
    PageFaultCount                    1086
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      294

        THREAD 81f351d8  Cid 0084.01b4  Teb: 7ffdd000 Win32Thread: e18721f0 WAIT: (Unknown)
UserMode Non-Alertable
            81fa4230  SynchronizationEvent
        Not impersonating
        DeviceMap                 e17a1a50
        Owning Process            8182aa20       Image:         calc.exe
        Attached Process          N/A            Image:         N/A
        Wait Start TickCount      341076         Ticks: 29 (0:00:00:00.290)
        Context Switch Count      177                LargeStack
        UserTime                  00:00:00.010
        KernelTime                00:00:00.030
        Win32 Start Address 0x010128a5
        Start Address 0x7c82b5c7
        Stack Init f808b000 Current f808abc4 Base f808b000 Limit f8086000 Call 0
        Priority 12 BasePriority 8 PriorityDecrement 2
        ChildEBP RetAddr
        f808abdc 80820128 nt!KiSwapContext+0x25 (FPO: [Uses EBP] [0,0,4])
        f808abf4 8081f9de nt!KiSwapThread+0x83 (FPO: [0,2,0])
        f808ac38 bf875ecb nt!KeWaitForSingleObject+0x2e0 (FPO: [5,12,4])
        f808ac94 bf87467d win32k!xxxSleepThread+0x1be (FPO: [SEH])
        f808acec bf87ad25 win32k!xxxRealInternalGetMessage+0x46a (FPO: [SEH])
        f808ad4c 8082337b win32k!NtUserGetMessage+0x3f (FPO: [SEH])
        f808ad4c 7c95ed54 nt!KiFastCallEntry+0xf8 (FPO: [0,0] TrapFrame @ f808ad64)
WARNING: Frame IP not in any known module. Following frames may be wrong.
        0007fdf4 00000000 0x7c95ed54

kd> !process 8182aa20 0
PROCESS 8182aa20  SessionId: 0  Cid: 0084    Peb: 7ffde000  ParentCid: 0668
    DirBase: 0ac9c000  ObjectTable: e1264a88  HandleCount:  41.
    Image: calc.exe
```

```
kd> !process 0n132 0
Searching for Process with Cid == 84
PROCESS 8182aa20  SessionId: 0  Cid: 0084    Peb: 7ffde000  ParentCid: 0668
    DirBase: 0ac9c000  ObjectTable: e1264a88  HandleCount:  41.
    Image: calc.exe
```

!process 0 0 xxx.exe 查找指定进程。

```
kd> !process 0 0 notepad.exe
PROCESS 81b7e5f0  SessionId: 0  Cid: 0184    Peb: 7ffd3000  ParentCid: 0668
    DirBase: 0a9a5000  ObjectTable: e104aab0  HandleCount:  41.
    Image: notepad.exe
```

!process -1 0 显示当前进程的基本信息。

```
kd> !process -1 0
PROCESS 808a0000  SessionId: none  Cid: 0000    Peb: 00000000  ParentCid: 0000
    DirBase: 00039000  ObjectTable: e1000d50  HandleCount: 203.
    Image: Idle

kd> !process 0 0 mspaint.exe
PROCESS 81de3be8  SessionId: 0  Cid: 03e8    Peb: 7ffd7000  ParentCid: 0788
    DirBase: 03284000  ObjectTable: e19b5d40  HandleCount:  74.
    Image: mspaint.exe

kd> .process /i 81de3be8
You need to continue execution (press 'g' <enter>) for the context
to be switched. When the debugger breaks in again, you will be in
the new process context.
kd> g
Break instruction exception - code 80000003 (first chance)
nt!RtlpBreakWithStatusInstruction:
8081d97e int     3
kd> !process -1 0
PROCESS 81de3be8  SessionId: 0  Cid: 03e8    Peb: 7ffd7000  ParentCid: 0788
    DirBase: 03284000  ObjectTable: e19b5d40  HandleCount:  74.
    Image: mspaint.exe
```

!process xxxxxxxx 2 显示指定进程的基本信息、线程列表以及每个线程的等待状态。

```
kd> !process 0 0 explorer.exe
PROCESS 81887020  SessionId: 0  Cid: 0668    Peb: 7ffdd000  ParentCid: 0658
    DirBase: 03ab2000  ObjectTable: e17743a0  HandleCount: 383.
    Image: Explorer.EXE

kd> !process 81887020 2
PROCESS 81887020  SessionId: 0  Cid: 0668    Peb: 7ffdd000  ParentCid: 0658
    DirBase: 03ab2000  ObjectTable: e17743a0  HandleCount: 383.
    Image: Explorer.EXE

        THREAD 81891db0  Cid 0668.066c  Teb: 7ffdf000 Win32Thread: e18759d0 WAIT: (Unknown) User
```

```
Mode Non-Alertable
          818823c8  SynchronizationEvent

       THREAD  8187ea38   Cid  0668.067c   Teb:  7ffdb000  Win32Thread:  e1892ea8  WAIT:  (Unknown)
UserMode Non-Alertable
          8187fd50  SynchronizationEvent

       THREAD  8187e288   Cid  0668.0680   Teb:  7ffda000  Win32Thread:  00000000  WAIT:  (Unknown)
UserMode Alertable
          8187e300  NotificationTimer

       THREAD  81878db0   Cid  0668.0688   Teb:  7ffd8000  Win32Thread:  00000000  WAIT:  (Unknown)
UserMode Alertable
          81883898  NotificationTimer
          8187f6d0  SynchronizationEvent
          81875848  NotificationEvent

       THREAD  8187b3c0   Cid  0668.0690   Teb:  7ffd7000  Win32Thread:  e18dc4f0  WAIT:  (Unknown)
UserMode Alertable
          8186f3ac  NotificationEvent
          81db2fbc  NotificationEvent
          8186c2ec  NotificationEvent
          818422f4  NotificationEvent
          818847c4  NotificationEvent
          81869c6c  NotificationEvent
          8188b14c  NotificationEvent
          8188c9d4  NotificationEvent
          8187e5a4  NotificationEvent
          81dd1bec  NotificationEvent
          8187f7b0  SynchronizationEvent

       THREAD  8185ddb0   Cid  0668.06d4   Teb:  7ff9f000  Win32Thread:  e1999878  WAIT:  (Unknown)
UserMode Non-Alertable
          818854e8  Semaphore Limit 0x7fffffff
          8185de28  NotificationTimer

       THREAD  81840458   Cid  0668.0714   Teb:  7ffd4000  Win32Thread:  e19755d8  WAIT:  (Unknown)
UserMode Non-Alertable
          818652c0  SynchronizationEvent

       THREAD  81828c78   Cid  0668.0720   Teb:  7ff9e000  Win32Thread:  e1a13648  WAIT:  (Unknown)
UserMode Non-Alertable
          8183d708  SynchronizationEvent

       THREAD  81dee420   Cid  0668.07f8   Teb:  7ffde000  Win32Thread:  e17b8008  WAIT:  (Unknown)
UserMode Non-Alertable
          818854e8  Semaphore Limit 0x7fffffff
          81dee498  NotificationTimer
```

```
      THREAD  81dd7388   Cid  0668.00dc   Teb:  7ffd6000  Win32Thread:  00000000  WAIT:  (Unknown)
UserMode Non-Alertable
          81dd7400  NotificationTimer

      THREAD  81f38a30   Cid  0668.0540   Teb:  7ffd9000  Win32Thread:  e19a8d38  WAIT:  (Unknown)
UserMode Non-Alertable
          8187b6d0  QueueObject
          81f38aa8  NotificationTimer
```

.process

.process 命令用来切换进程上下文，比如显示某个进程内存时，就需要先切换到该进程，然后再显示内存。

不指定参数，.process 命令切换回中断时的进程上下文。

```
kd> .process
Implicit process is now 808a0000
```

.process /r /p xxxxxxxx 切换到指定进程上下文，接下来的命令都会使用新进程的上下文，比如进程内存。但这只是影响 WinDbg 命令的输出，没有改变目标系统。

```
kd> .process
Implicit process is now 808a0000
kd> db 1000000 l60
01000000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
kd> !process 0 0 calc.exe
PROCESS 8182aa20  SessionId: 0  Cid: 0084    Peb: 7ffde000  ParentCid: 0668
    DirBase: 0ac9c000  ObjectTable: e1264a88  HandleCount:  41.
    Image: calc.exe

kd> .process /r /p 8182aa20
Implicit process is now 8182aa20
.cache forcedecodeuser done
Loading User Symbols
kd> db 1000000 l60
01000000  4d 5a 90 00 03 00 00 00-04 00 00 00 ff ff 00 00  MZ..............
01000010  b8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  ........@.......
01000020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
01000030  00 00 00 00 00 00 00 00-00 00 00 00 e8 00 00 00  ................
01000040  0e 1f ba 0e 00 b4 09 cd-21 b8 01 4c cd 21 54 68  ........!..L.!Th
01000050  69 73 20 70 72 6f 67 72-61 6d 20 63 61 6e 6e 6f  is program canno
```

.process /i xxxxxxxx 切换到指定进程上下文，和.process /r /p 命令不同的是，该命令会指示目标操作系统做实际的进程切换，所以该命令执行后还需要 g 一下，让目标系统有机会切换进程，然后会再次中断到 WinDbg 中。该

命令只能在 WindowsXP 及之后的系统上使用。

```
kd> .process
Implicit process is now 808a0000
kd> db 1000000 l60
01000000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
01000050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
kd> !process 0 0 calc.exe
PROCESS 8182aa20  SessionId: 0  Cid: 0084    Peb: 7ffde000  ParentCid: 0668
    DirBase: 0ac9c000  ObjectTable: e1264a88  HandleCount:  41.
    Image: calc.exe

kd> .process /i 8182aa20
You need to continue execution (press 'g' <enter>) for the context
to be switched. When the debugger breaks in again, you will be in
the new process context.
kd> g
Break instruction exception - code 80000003 (first chance)
nt!RtlpBreakWithStatusInstruction:
8081d97e int     3
kd> db 1000000 l60
01000000  4d 5a 90 00 03 00 00 00-04 00 00 00 ff ff 00 00  MZ..............
01000010  b8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  ........@.......
01000020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
01000030  00 00 00 00 00 00 00 00-00 00 00 00 e8 00 00 00  ................
01000040  0e 1f ba 0e 00 b4 09 cd-21 b8 01 4c cd 21 54 68  ........!..L.!Th
01000050  69 73 20 70 72 6f 67 72-61 6d 20 63 61 6e 6e 6f  is program canno
```

!dml_proc

!dml_proc 也用来显示进程信息，不带参数时直接显示进程列表，比!process 0 0 命令显示的格式要紧凑好看点。

```
kd> !dml_proc
Address  PID  Image file name
81f9b9e8 4    System
81daa6d0 148  SMSS.EXE
81da61d0 180  csrss.exe
81f12d88 198  winlogon.exe
81e97998 1c4  services.exe
81d576c8 1d0  lsass.exe
81e95d88 27c  svchost.exe
81eedd88 2d4  svchost.exe
81f06bc8 30c  svchost.exe
81db7a08 324  svchost.exe
```

```
81e96848 344  svchost.exe
81f056e8 3bc  spoolsv.exe
81d71990 3e4  msdtc.exe
81d59b18 444  vmsrvc.exe
81d9cd88 464  svchost.exe
81e5a938 49c  svchost.exe
818ead88 514  vpcmap.exe
81e808c8 578  svchost.exe
81887020 668  Explorer.EXE
81856688 6d0  vmusrvc.exe
818572e0 6e4  ctfmon.exe
8183c408 740  wmiprvse.exe
81f30d88 320  conime.exe
```

线程相关

线程操作包括：显示线程信息，切换线程上下文等。

!thread

!thread 命令显示线程详细信息，包括线程 ETHREAD、线程 ID、等待状态、起始地址、当前堆栈等。

!thread 或!thread -1 命令显示当前线程的详细信息。

```
kd> !thread -1
THREAD 81f97020  Cid 0004.004c  Teb: 00000000 Win32Thread: 00000000 RUNNING on processor 0
Not impersonating
DeviceMap                 e136d238
Owning Process            81f9b9e8       Image:         System
Attached Process          81de3be8       Image:         mspaint.exe
Wait Start TickCount      106651         Ticks: 0
Context Switch Count      2398
UserTime                  00:00:00.000
KernelTime                00:00:00.080
Start Address nt!ExpWorkerThread (0x80820300)
Stack Init f88af000 Current f88aed00 Base f88af000 Limit f88ac000 Call 0
Priority 12 BasePriority 12 PriorityDecrement 0
ChildEBP RetAddr  Args to Child
f88aed2c 809e7bb3 00000007 81f97020 808b059c nt!RtlpBreakWithStatusInstruction (FPO: [1,0,0])
f88aed80 808203bd 00000000 00000000 81f97020 nt!ExpDebuggerWorker+0xac (FPO: [SEH])
f88aedac 80905d2c 00000000 00000000 00000000 nt!ExpWorkerThread+0xeb (FPO: [1,5,0])
f88aeddc 80828499 80820300 00000001 00000000 nt!PspSystemThreadStartup+0x2e (FPO: [SEH])
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
```

!thread -t xxx 显示指定线程的信息，xxx 表示线程 ID。

```
kd> !thread -t 4c
Looking for thread Cid = 4c ...
THREAD 81f97020  Cid 0004.004c  Teb: 00000000 Win32Thread: 00000000 RUNNING on processor 0
```

```
Not impersonating
DeviceMap                 e136d238
Owning Process            81f9b9e8       Image:         System
Attached Process          81de3be8       Image:         mspaint.exe
Wait Start TickCount      106651         Ticks: 0
Context Switch Count      2398
UserTime                  00:00:00.000
KernelTime                00:00:00.080
Start Address nt!ExpWorkerThread (0x80820300)
Stack Init f88af000 Current f88aed00 Base f88af000 Limit f88ac000 Call 0
Priority 12 BasePriority 12 PriorityDecrement 0
ChildEBP RetAddr  Args to Child
f88aed2c 809e7bb3 00000007 81f97020 808b059c nt!RtlpBreakWithStatusInstruction (FPO: [1,0,0])
f88aed80 808203bd 00000000 00000000 81f97020 nt!ExpDebuggerWorker+0xac (FPO: [SEH])
f88aedac 80905d2c 00000000 00000000 00000000 nt!ExpWorkerThread+0xeb (FPO: [1,5,0])
f88aeddc 80828499 80820300 00000001 00000000 nt!PspSystemThreadStartup+0x2e (FPO: [SEH])
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
```

!thread xxxxxxxx 16 显示指定线程的信息，0x16 是输出掩码，不指定掩码时缺省为 6，添加 0x10 表示输出线程信息前先切换到线程所属进程的上下文，这样显示的线程堆栈更精确。从下表能看到这一点，后面一个堆栈才是正确堆栈，正好是 notepad.exe 进程的堆栈。而且两个堆栈前面部分相同，从 ring3 开始的部分就完全不同，这是因为进程上下文不一样导致的。

```
kd> !process 0 0 notepad.exe
PROCESS 81e77170  SessionId: 0  Cid: 039c    Peb: 7ffdf000  ParentCid: 0788
    DirBase: 1b42f000  ObjectTable: e19b4008  HandleCount:  41.
    Image: notepad.exe

kd> !process 81e77170 2
PROCESS 81e77170  SessionId: 0  Cid: 039c    Peb: 7ffdf000  ParentCid: 0788
    DirBase: 1b42f000  ObjectTable: e19b4008  HandleCount:  41.
    Image: notepad.exe

        THREAD 81e28020   Cid 039c.03d0   Teb: 7ffde000 Win32Thread: e1a333d8 WAIT: (Unknown)
UserMode Non-Alertable
          8187cf30  SynchronizationEvent


kd> !thread 81e28020
THREAD 81e28020  Cid 039c.03d0  Teb: 7ffde000 Win32Thread: e1a333d8 WAIT: (Unknown) UserMode
Non-Alertable
    8187cf30  SynchronizationEvent
Not impersonating
DeviceMap                 e136d238
Owning Process            81e77170       Image:         notepad.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      106196         Ticks: 455 (0:00:00:04.556)
Context Switch Count      400                LargeStack
UserTime                  00:00:00.010
```

```
KernelTime              00:00:00.100
Win32 Start Address mspaint!CProgressCtrl::`vftable' (0x010073a5)
Start Address kernel32!BaseProcessStartThunk (0x7c82b5c7)
Stack Init f50fe000 Current f50fdbc4 Base f50fe000 Limit f50f9000 Call 0
Priority 10 BasePriority 8 PriorityDecrement 0
ChildEBP RetAddr  Args to Child
f50fdbdc 80820128 81e28020 81e280c8 00000000 nt!KiSwapContext+0x25 (FPO: [Uses EBP] [0,0,4])
f50fdbf4 8081f9de 00000000 e1a333d8 00000000 nt!KiSwapThread+0x83 (FPO: [0,2,0])
f50fdc38 bf875ecb 8187cf30 0000000d 00000001 nt!KeWaitForSingleObject+0x2e0 (FPO: [5,12,4])
f50fdc94 bf87467d 000025ff 00000000 00000001 win32k!xxxSleepThread+0x1be (FPO: [SEH])
f50fdcec  bf87ad25  f50fdd18  00000000  00000000  win32k!xxxRealInternalGetMessage+0x46a  (FPO:
[SEH])
f50fdd4c 8082337b 0007fefc 00000000 00000000 win32k!NtUserGetMessage+0x3f (FPO: [SEH])
f50fdd4c 7c95ed54 0007fefc 00000000 00000000 nt!KiFastCallEntry+0xf8 (FPO: [0,0] TrapFrame @
f50fdd64)
0007feb8 77e2c7c0 0103ed8c 00000000 00000000 ntdll!KiFastSystemCallRet (FPO: [0,0,0])
0007fed8 0103ed8c 00000000 00000000 00000000 USER32!GetMessageW+0x33 (FPO: [4,0,4])
0007fef0 7d262d4a 0103ed58 0103ed58 ffffffff mspaint!theApp+0x34
WARNING: Stack unwind information not available. Following frames may be wrong.
0007ff08 7d234ceb ffffffff 00000002 7ffd7000 MFC42u!Ordinal5711+0x4a
0007ff1c 01034e8e 01000000 00000000 0002061c MFC42u!Ordinal1569+0x7b
0007ffc0 7c8123cd 00000000 00000000 7ffd7000 mspaint!wWinMainCRTStartup+0x199 (FPO: [SEH])
0007fff0 00000000 01034cf5 00000000 78746341 kernel32!BaseProcessStart+0x23 (FPO: [SEH])

kd> !thread 81e28020 16
THREAD 81e28020  Cid 039c.03d0  Teb: 7ffde000 Win32Thread: e1a333d8 WAIT: (Unknown) UserMode
Non-Alertable
    8187cf30  SynchronizationEvent
Not impersonating
DeviceMap               e136d238
Owning Process          81e77170      Image:         notepad.exe
Attached Process        N/A           Image:         N/A
Wait Start TickCount    106196        Ticks: 455 (0:00:00:04.556)
Context Switch Count    400               LargeStack
UserTime                00:00:00.010
KernelTime              00:00:00.100
Win32 Start Address notepad!WinMainCRTStartup (0x010073a5)
Start Address kernel32!BaseProcessStartThunk (0x7c82b5c7)
Stack Init f50fe000 Current f50fdbc4 Base f50fe000 Limit f50f9000 Call 0
Priority 10 BasePriority 8 PriorityDecrement 0

ChildEBP RetAddr  Args to Child
f50fdbdc 80820128 81e28020 81e280c8 00000000 nt!KiSwapContext+0x25 (FPO: [Uses EBP] [0,0,4])
f50fdbf4 8081f9de 00000000 e1a333d8 00000000 nt!KiSwapThread+0x83 (FPO: [0,2,0])
f50fdc38 bf875ecb 8187cf30 0000000d 00000001 nt!KeWaitForSingleObject+0x2e0 (FPO: [5,12,4])
f50fdc94 bf87467d 000025ff 00000000 00000001 win32k!xxxSleepThread+0x1be (FPO: [SEH])
f50fdcec bf87ad25 f50fdd18 00000000 00000000 win32k!xxxRealInternalGetMessage+0x46a(FPO: [SEH])
f50fdd4c 8082337b 0007fefc 00000000 00000000 win32k!NtUserGetMessage+0x3f (FPO: [SEH])
```

```
f50fdd4c 7c95ed54 0007fefc 00000000 00000000 nt!KiFastCallEntry+0xf8 (FPO: [0,0] TrapFrame @
f50fdd64)
0007feb8 77e2c78d 77e2c7c0 0007fefc 00000000 ntdll!KiFastSystemCallRet (FPO: [0,0,0])
0007fed8 01002a3b 0007fefc 00000000 00000000 USER32!NtUserGetMessage+0xc
0007ff1c 0100752b 01000000 00000000 000a24b2 notepad!WinMain+0xe5 (FPO: [4,8,0])
0007ffc0 7c8123cd 00000000 00000000 7ffdf000 notepad!WinMainCRTStartup+0x186 (FPO: [SEH])
0007fff0 00000000 010073a5 00000000 78746341 kernel32!BaseProcessStart+0x23 (FPO: [SEH])
```

.thread

.thread 命令设置线程上下文，线程上下文包括寄存器值、堆栈等。不指定参数时，.thread 切换回中断时的线程上下文。

```
kd> .thread
Implicit thread is now 81f97020
kd> !thread -p -1 0
PROCESS 81de3be8  SessionId: 0  Cid: 03e8    Peb: 7ffd7000  ParentCid: 0788
    DirBase: 03284000  ObjectTable: e19b5d40  HandleCount:  74.
    Image: mspaint.exe

THREAD 81f97020  Cid 0004.004c  Teb: 00000000 Win32Thread: 00000000 RUNNING on processor 0
kd> r
eax=00000007 ebx=00000000 ecx=00000000 edx=00000000 esi=81de3be8 edi=00000000
eip=8081d97e esp=f88aed30 ebp=f88aed80 iopl=0         nv up ei pl zr na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000246
nt!RtlpBreakWithStatusInstruction:
8081d97e int     3
kd> k
ChildEBP RetAddr
f88aed2c 809e7bb3 nt!RtlpBreakWithStatusInstruction
f88aed80 808203bd nt!ExpDebuggerWorker+0xac
f88aedac 80905d2c nt!ExpWorkerThread+0xeb
f88aeddc 80828499 nt!PspSystemThreadStartup+0x2e
00000000 00000000 nt!KiThreadStartup+0x16
kd> .thread /p /r 81e28020
Implicit thread is now 81e28020
Implicit process is now 81e77170
.cache forcedecodeuser done
Loading User Symbols
.....................
kd> r
Last set context:
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
eip=808207bc esp=f50fdbd0 ebp=f50fdbf4 iopl=0         nv up di pl nz na po nc
cs=0008  ss=0010  ds=0000  es=0000  fs=0000  gs=0000             efl=00000000
nt!KiSwapContext+0x25:
808207bc mov     ebp,dword ptr [esp]                ss:0010:f50fdbd0=f50fdbf4
kd> k
```

```
  *** Stack trace for last set context - .thread/.cxr resets it
ChildEBP RetAddr
f50fdbdc 80820128 nt!KiSwapContext+0x25
f50fdbf4 8081f9de nt!KiSwapThread+0x83
f50fdc38 bf875ecb nt!KeWaitForSingleObject+0x2e0
f50fdc94 bf87467d win32k!xxxSleepThread+0x1be
f50fdcec bf87ad25 win32k!xxxRealInternalGetMessage+0x46a
f50fdd4c 8082337b win32k!NtUserGetMessage+0x3f
f50fdd4c 7c95ed54 nt!KiFastCallEntry+0xf8
0007feb8 77e2c78d ntdll!KiFastSystemCallRet
0007fed8 01002a3b USER32!NtUserGetMessage+0xc
0007ff1c 0100752b notepad!WinMain+0xe5
0007ffc0 7c8123cd notepad!WinMainCRTStartup+0x186
0007fff0 00000000 kernel32!BaseProcessStart+0x23
```