Topics

• Catching data corruption
• Software and hardware breakpoints
• Breakpoint strategy
• Disabling and enabling breakpoints
// Worker thread periodically writes a command to a controller
// The controller is implemented as an address in memory
// Corruption of controller memory value is modelled as an exception

DWORD WINAPI WorkItemNormal(LPVOID lpParameter)
{
    Controller = COMMAND;

    while(1)
    {
        Sleep(WAIT);
        if (Controller != COMMAND)
        {
            RaiseException(0xBADC, 0, 1, (const ULONG_PTR *)&Controller);
        }
        Controller = COMMAND; // we remember the address of this instruction
    }
}
// Enable hardware write access breakpoint

DWORD WINAPI WorkItemNormal(LPVOID lpParameter)
{
    Controller = COMMAND; // expect a breakpoint hit here, ignore

    while(1)
    {
        Sleep(WAIT);
        if (Controller != COMMAND)
        {
            RaiseException(0xBADC, 0, 1, (const ULONG_PTR *)&Controller);
        }
        // Disable hardware write access breakpoint, ignore
        Controller = COMMAND;
        // Enable hardware write access breakpoint, resume
    }
}
Debugger Output

0:001> ba w4 Controller

0:001> bp MixedBreakpoints!WorkItemNormal+0x4a "bd 0; t" * disable, skip

0:001> u MixedBreakpoints!WorkItemNormal+0x4a
MixedBreakpoints!WorkItemNormal+0x4a:
00000001`3fe414ea c705ecaa000010000000 mov dword ptr [MixedBreakpoints!Controller (00000001`3fe4bfe0)],10h
00000000`3fe414f4 ebbd [...] 

0:001> bp 00000001`3fe414f4 "be 0; g" * enable, resume

0:001> bl
0 e 00000001`3f03bfe0 w 4 0001 (0001) 0:**** MixedBreakpoints!Controller
1 e 00000001`3f0314ea 0001 (0001) 0:**** MixedBreakpoints!WorkItemNormal+0x4a "bd 0; t"
2 e 00000001`3f0314f4 0001 (0001) 0:**** MixedBreakpoints!WorkItemNormal+0x54 "be 0; g"

0:001> g; g * we skip the first write
Breakpoint 0 hit
Breakpoint 0 hit
MixedBreakpoints!WorkItemDefect+0x12:
00000001`3fdc1512 33c0 xor eax,eax

0:002> k
Child-SP RetAddr Call Site
00000000`02f8fda8
00000000`02f8fda0 00000000`76d5c521 kernel32!BaseThreadInitThunk+0xd
00000000`02f8fde0 00000000`00000000 ntdll!RtlUserThreadStart+0x1d

© 2012 DumpAnalysis.org + TraceAnalysis.org
Commands

ba - set hardware access breakpoint
bp - set software code breakpoint
bl - list breakpoints
bd - disable breakpoint
be - enable breakpoint
t - trace one instruction
April 2, 2012: **Introduction to Software Narratology** (free)

April 11-16, 2012: **Accelerated Windows Memory Dump Analysis**

April 20-23, 2012: **Advanced Windows Memory Dump Analysis**

April 27-30, 2012: **Accelerated Software Trace Analysis**

Forthcoming: **Accelerated Mac OS X Core Dump Analysis**

**Training Schedule**
Debugging.TV