

Driver Debugging Basics

Khalil Nassar
Senior Systems Engineer
Microsoft Corporation

Agenda

- Debug 01100101 (Debug 101)
- x64 versus x86 Differences
- Essential Command Reference
- Windows Vista and Windows Server code name “Longhorn” Architectural Changes
- Debugging Techniques
- Top 10 Questions

Debug 01100101

- What is a debugger
- State
- Debugger view
- Breakpoints and scripts

Debug 01100101

- What is a debugger?

Debug 01100101

- State

Virtual memory

Process List -> Directory Base List -> VM
-> Thread List -> Reg

Context

Debug 01100101

- State

Interrupts

Timeslice/Dispatch

demo

Manipulating Debugger View

Debug 01100101

Manipulating Debugger View

- .process
- .cxr, .trap
- .thread
- .frame

Debug 01100101

Breakpoints and Scripts

- Pseudo registers
- Aliases

Debug 01100101

Useful Pseudo Registers

- \$teb,\$peb,\$p,\$ea,\$proc,\$thread,\$tid,\$tpid,\$mod,\$base,\$addr,\$imagename
- \$ea2 – for instructions that have 2 effective addresses
- \$callret
- \$dbgtime – debugger's current time
- \$scopeip – returns the instruction pointer for the currently set scope
- \$bp – last hit break point

Debug 01100101

- \$bphit – user ID of breakpoint just hit
- \$frame – current frame number
- \$! – prefixing a symbol with \$! will cause only the current scope to be searched
- \$exentry – address of the entry point for the first executable of the current process
- \$t0 - \$t9 – actual pseudo registers used for temporary values

Debug 01100101

- \$ip – The current instruction pointer
- \$eventip - The IP at the time of the current event. This can be different from \$ip if you switch threads or manually change the IP register
- \$previp - The \$eventip value from the last event. The last event for a user means the last prompt. If there wasn't a last event it'll be an error
- \$relip - Any related IP value for the current event. When you are branch tracing it'll be the branch source, otherwise it'll be an error.
- \$retreg = eax (x86) , ret0 (ia64),rax (x64)
- \$CurrentDumpPath

Debug 01100101

Debugger Aliases

- @#ModuleName string
- @#ImageName string
- @#LoadedImageName string
- @#SymbolFileName string
- @#MappedImageName string

Debug 01100101

- @#Base ULONG64
- @#Size ULONG
- @#TimeStamp ULONG
- @#Checksum ULONG
- @#Flags ULONG
- @#SymbolType USHORT

Debug 01100101

- @#ImageNameSize ULONG
- @#ModuleNameSize ULONG
- @#LoadedImageNameSize ULONG
- @#SymbolFileNameSize ULONG
- @#MappedImageNameSize ULONG

Debug 01100101

Useful Breakpoints

- Self clearing call returning –
`bp func "bp /1 @$ra \"r$retreg\";g"`
- Set a bp on a yet to be defined module –
`bu /1 wmain "ba w4 g_Var \"j (
 @@(g_Var==%1)) '.echo broken
 because g_Var is %1'; 'gc' \";g"`
`bu notepad!winmain ".printf
 \"notepad!winmain entered with hInstance
 = %p\\n\", poi(hInstance);g"`

Debug 01100101

Script Examples

- Search for kernel trap frames.
Demonstrates arbitrary processing on each hit

```
.foreach ($addr { s-[1]d 80000000 |?7fffffff  
23 23 }) { ? $addr ; .trap $addr - 0n52 ; kv  
}!vm
```

Debug 01100101

Script Examples continued

Display full callstack for all threads

```
r? $t0 = &nt!PsActiveProcessHead
.for (r $t1 = poi(@$t0); (@$t1 != 0) & (@$t1 != @$t0); r $t1 =
    poi(@$t1))
{
    r? $t2 = #CONTAINING_RECORD(@$t1,
    nt!_EPROCESS, ActiveProcessLinks);
    .process /p /r $t2
    !process $t2 7
}
```

x64 And x86 Differences

- Architectural Issues
- Debugging Relevant Issues

x64 And x86 Differences

Architectural

- Registers
- Exception handling
- Stack walking

x64 And x86 Differences

Debugging Relevant

- Debug 32-bit processes with the 32-bit debugger
- UMDH issues with x64
- From the debugger – access CPU registers with @
- Issues encountered building the Keyboard filter driver for x64
- Virtual memory translation
- Practice inspection with quad words (dq)

x64 And x86 Differences

Trap Frames

- Nonvolatile registers (rbx, rsi, rdi, etc.) not preserved for perf reasons
- Must dig them out of the callee stacks

Essential Commands

Debugger Setup

- `.sympath`, `.srcpath`, `.lsrspath`, `.lines`
- `.reload`, `!ml`
- `!sym noisy`
- `.enable_unicode 1`
- `x`

Essential Commands - 2

Virtual Memory

- !pool, !poolused, !poolval, !poolfind
- !vm
- !vprot, !address

Essential Commands - 3

System Wide

- !locks, !irpfind -4/v
- !pcr, !idt
- !object, !drvobj, !devstack
- !cpuid

Essential Commands - 4

Relative To Current Thread

- !peb, !teb
- !handle
- !thread
- .cxr, .trap
- .exr -1

Essential Commands - 5

Relative To Current Process

- !process, !pcr

Essential Commands - 6

Error Analysis

- !analyze -v, !verifier, !avrf
- !error, !errorlog, !gle
- .exr -1, .eventlog, .lastevent

Essential Commands - 7

Data Analysis

- dv, dt, ?, ??
- k (kp, kP, kv, kn)
- r (rMff)
- .formats
- d (dc, du, dq, dl, dds, dqs)
- !d
- u, ub, uf

Essential Commands - 8

Execution

- g, t, p, wt, bp, bu
- SX

New Commands

- `.bpsync 1`
- `.flash_on_break`

Windows Vista/Windows Server Longhorn Architectural Changes

- Improved Thread Pooling – including multiple thread pools
- Boot environment reengineered
- Need KD for unsigned kernel drivers on x64

demo

Debugging Techniques

Top 10 Debugger Questions

- #10: Is there a way to redirect the output of a debugger extension to a text file?

Top 10 Debugger Questions

- #9: Is there a way to make the debugger flash or emit a sound when a breakpoint is hit?

Top 10 Debugger Questions

● #8: .kdfiles on Windows Vista

Top 10 Debugger Questions

- #7: Breaking in Main() from KD.
Module Load

Top 10 Debugger Questions

• #6: .crash behavior

Top 10 Debugger Questions

● #5: BCDEDIT

Top 10 Debugger Questions

● #4: Why Does KD Get Wedged

Top 10 Debugger Questions

• #3: kd -kl

Top 10 Debugger Questions

• #2: Can I force the symbols to match?

Top 10 Debugger Questions



Why is the sky blue?

Summary

- Debugging effectively requires understanding your target code, debugger theory and Operating System (OS) theory. This presentation has been an introduction to operating system and debugger theory with an emphasis on debugger capabilities and commands. The related lab gives hands on experience with driver and OS theory using the debugger as the enabler

Call To Action

- Understand the system state, not just your driver
 - Virtual Memory
 - Interrupts
 - Timeslice/Dispatch
- Go beyond Call Stacks and Exceptions
 - Know more of the essential commands
- Debugging Well is Very Rewarding

Additional Resources

- Web Resources

- Debugging Tools for Windows:

<http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx>

- Training, message boards, etc:

<http://www.microsoft.com/whdc/devtools/debugging/resources.mspx>

- Related Sessions

- DVR-T410 Driver Debugging Basics

- DVR-C478 Debugging Drivers: Discussion

- DVR-C408 Driver Verifier: Internals Discussion

- DVR-H409 Debugging Bugs Exposed by Driver Verifier: Workshop

- DVR-H481 64-bit Driver Debugging Basics: Workshop (2 sessions)

- Help: Create a support incident: [DDK Developer Support](#)

- Feedback: Send suggestions or bug reports:

[Windbgfb @ microsoft.com](mailto:Windbgfb@microsoft.com)

Microsoft[®]

Your potential. Our passion.[™]

© 2007 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

