使用DCI EXDI会话调试Windows内核

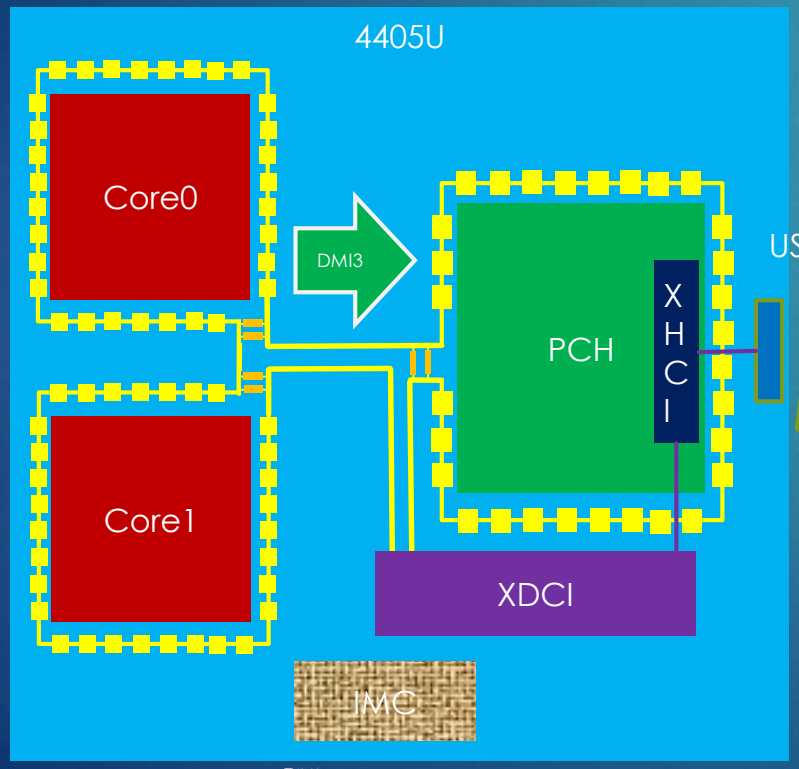
张银奎

2020/6/6 上海

张银奎，Raymond Zhang，格蠹老雷，《软件调试》和《格蠹汇编》作者
http://advdbg.org http://weibo.com/dbgger 格友公众号


工作原理
两种前端
调试NT内核
调试BootMgr
高级命令

调试模型



USB3.0

DCI

USB3.0接口

OpenIPC_x64.exe

WinDBG/NanoCode

DbgEng.dll

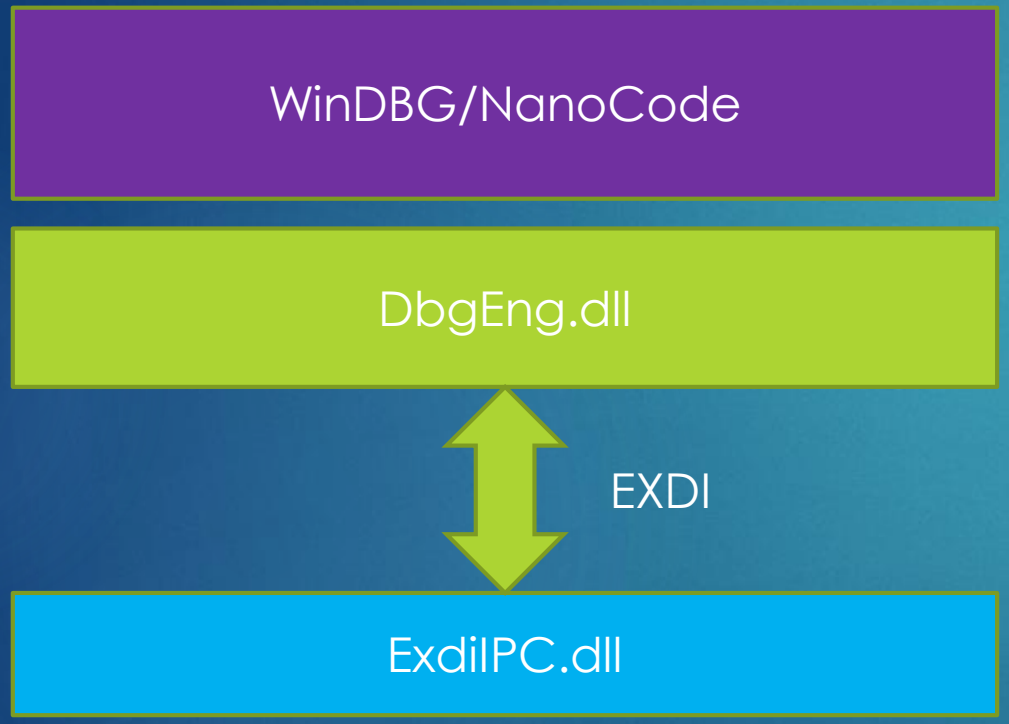
ExdiIPC.dll

Dciusb.sys

X86硬件

调试主机

EXDI



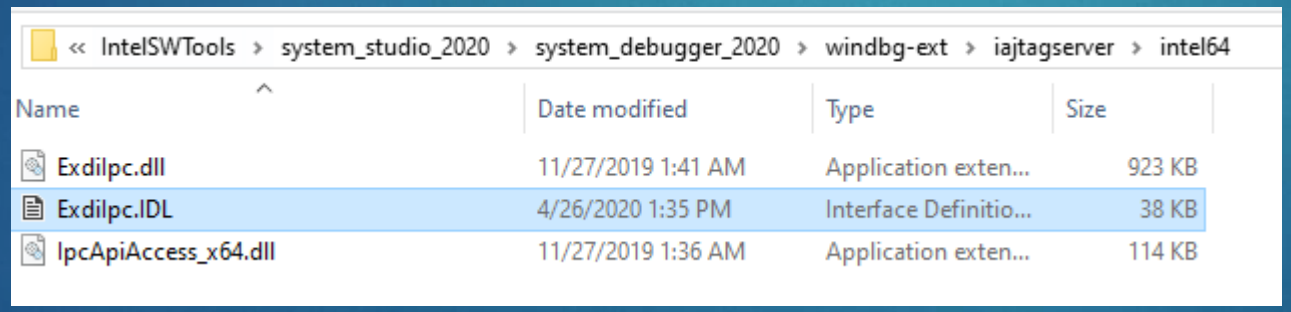
- eXtended Debug Interface
- DbgEng与外部调试器的接口
- 最初版本似乎为1999年
  - Greg Hogdal 11-Nov-1999
- 目前使用的v3版本
  - Ivan Shcherbakov (ivshcher) 31-Oct-2013

IDL头信息

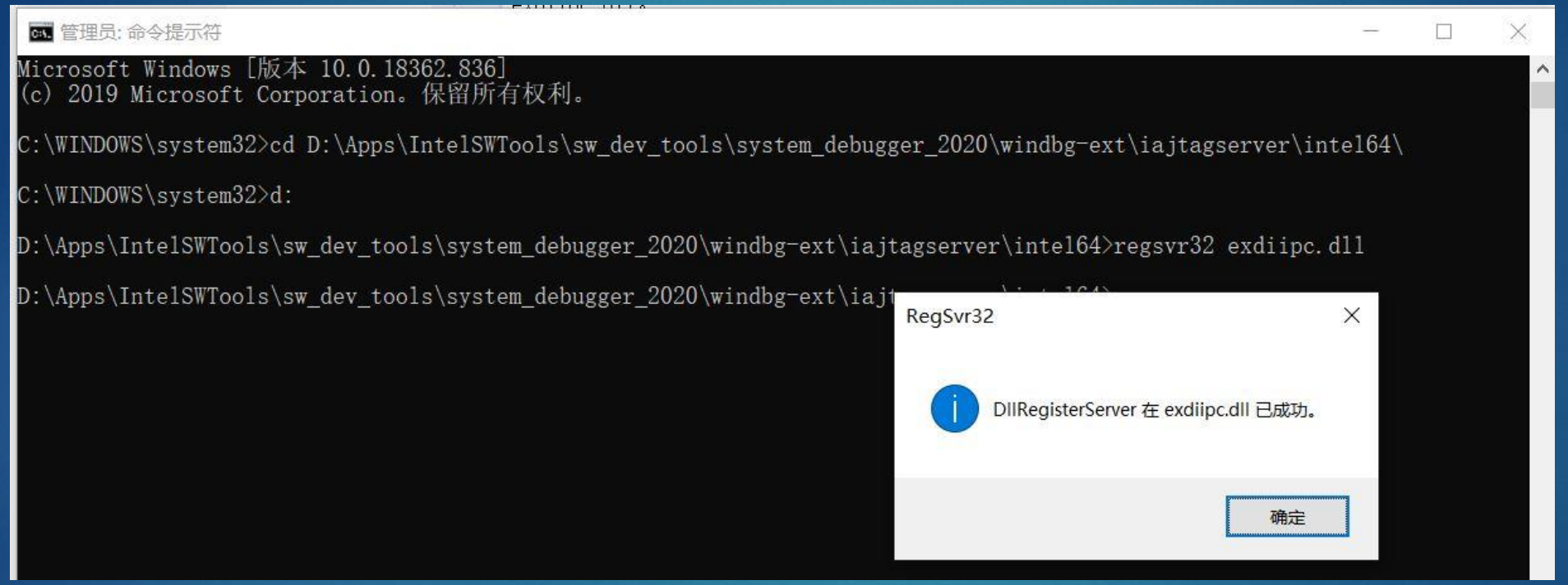
```
Module Name:

    eXdi3.idl : IDL source for eXdi v3 interface

Abstract:

    This file will be processed by the MIDL tool to
    produce the type library and marshalling code.

    eXDI: eXtended Debug Interface.
    The interfaces described here are meant to be used for communication between an external debug driver
    (like driver for Hardware Debug probe).


Author:

    Greg Hogdal 11-Nov-1999
    Ivan Shcherbakov (ivshcher) 31-Oct-2013

Environment:

    Win32 NT
```

EXDIIPC.dll

- 英特尔System Studio中的模块
- COM组件
- 未开源，没有公开PDB
- 安装ISS时如果没有检测到系统里有WinDBG，可能不安装

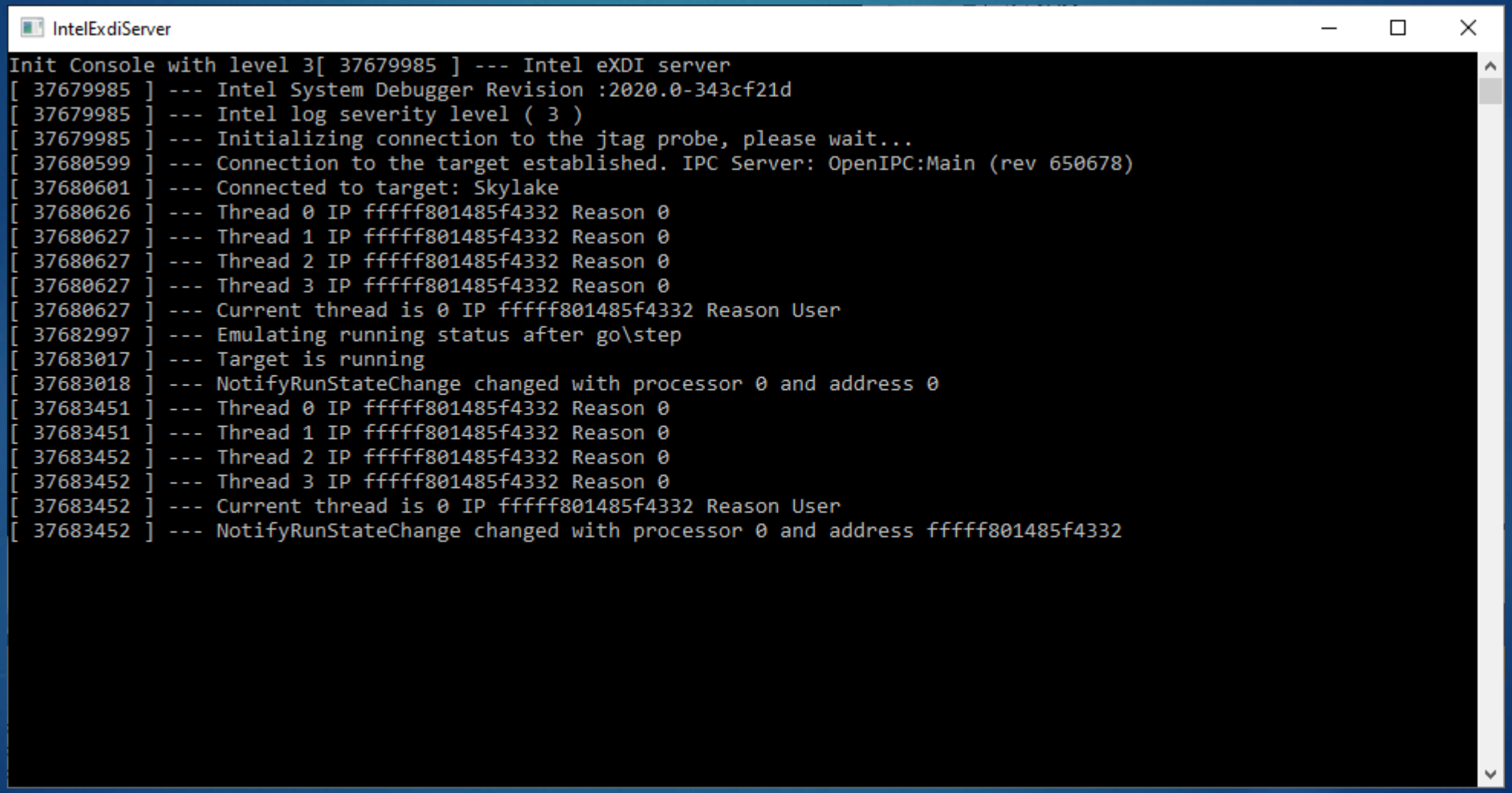


手工注册



> regsvr32 exdiipc.dll

工作中


IntelExdiServer
Init Console with level 3[37679985] --- Intel eXDI server
[37679985] --- Intel System Debugger Revision :2020.0-343cf21d
[37679985] --- Intel log severity level (3)
[37679985] --- Initializing connection to the jtag probe, please wait...
[37680599] --- Connection to the target established. IPC Server: OpenIPC:Main (rev 650678)
[37680601] --- Connected to target: Skylake
[37680626] --- Thread 0 IP fffff801485f4332 Reason 0
[37680627] --- Thread 1 IP fffff801485f4332 Reason 0
[37680627] --- Thread 2 IP fffff801485f4332 Reason 0
[37680627] --- Thread 3 IP fffff801485f4332 Reason 0
[37680627] --- Current thread is 0 IP fffff801485f4332 Reason User
[37682997] --- Emulating running status after go\step
[37683017] --- Target is running
[37683018] --- NotifyRunStateChange changed with processor 0 and address 0
[37683451] --- Thread 0 IP fffff801485f4332 Reason 0
[37683451] --- Thread 1 IP fffff801485f4332 Reason 0
[37683452] --- Thread 2 IP fffff801485f4332 Reason 0
[37683452] --- Thread 3 IP fffff801485f4332 Reason 0
[37683452] --- Current thread is 0 IP fffff801485f4332 Reason User
[37683452] --- NotifyRunStateChange changed with processor 0 and address fffff801485f4332

进程外的COM组件

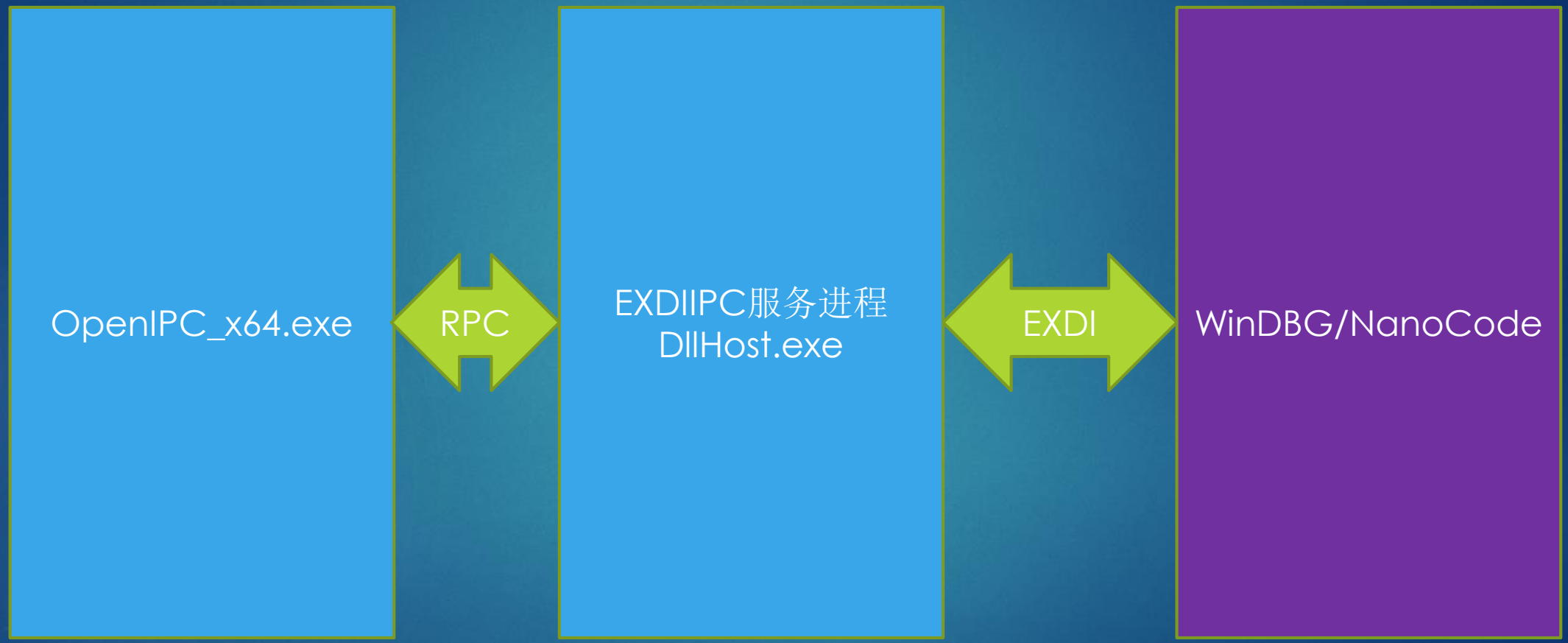
```
module name
DllHost     (pdb symbols)          C:\ProgramData\dbg\sym\dllhost.pdb\DCB6DFC011D09009B77832319E29E9051\dllhost.pdb
StructuredData x64    (deferred)
Remote_Ipc_x64    (export symbols)       C:\IntelSWTools\system_studio_2020\tools\OpenIPC_1.2011.4508.100\bin\Remote_Ipc_x64.dll
Logging_x64    (deferred)
ExdiIpc     (export symbols)       C:\NanoCode\exts\ExdiIpc.dll
MSVCP140    (private pdb symbols)  C:\ProgramData\dbg\sym\msvcp140.amd64.pdb\229884A55A924F948B58DA34C99BE8F71\msvcp140.amd64.pdb
Common_x64    (deferred)
IpcApi_x64    (deferred)
IpcApiImpl_x64    (deferred)
Error_x64    (deferred)
IpcApiLogging_x64    (deferred)
IpcApiAccess_x64    (deferred)
```

与OpenIPC通信

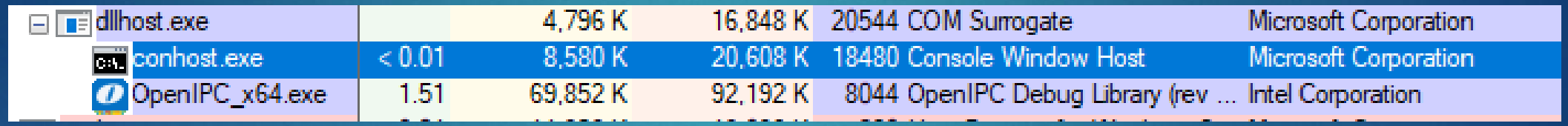
- RPC
- 用网络做传输层

```
Call Site
ntdll!NtWaitForMultipleObjects+0x14
KERNELBASE!WaitForMultipleObjectsEx+0x107
combase!MTAThreadWaitForCall+0x115 [onecore\com\combase\dcomrem\channelb.cxx @ 7140]
combase!MTAThreadDispatchCrossApartmentCall+0xc5 [onecore\com\combase\dcomrem\chancont.cxx @ 229]
combase!CSyncClientCall::SwitchAptAndDispatchCall+0x325 [onecore\com\combase\dcomrem\channelb.cxx @ 5696]
combase!CSyncClientCall::SendReceive2+0x407 [onecore\com\combase\dcomrem\channelb.cxx @ 5377]
combase!SyncClientCallRetryContext::SendReceiveWithRetry+0x25 [onecore\com\combase\dcomrem\callctrl.cxx @ 1617]
combase!CSyncClientCall::SendReceiveInRetryContext+0x25 [onecore\com\combase\dcomrem\callctrl.cxx @ 567]
combase!DefaultSendReceive+0x64 [onecore\com\combase\dcomrem\callctrl.cxx @ 525]
combase!CSyncClientCall::SendReceive+0x128 [onecore\com\combase\dcomrem\ctxchnl.cxx @ 783]
combase!CClientChannel::SendReceive+0x84 [onecore\com\combase\dcomrem\ctxchnl.cxx @ 653]
combase!NdrExtpProxySendReceive+0x4e [onecore\com\combase\ndr\ndrole\proxy.cxx @ 2002]
RPCRT4!NdrpClientCall2+0x40a
combase!ObjectStublessClient+0x1df [onecore\com\combase\ndr\ndrole\amd64\stblsclt.cxx @ 368]
combase!ObjectStubless+0x42 [onecore\com\combase\ndr\ndrole\amd64\stubless.asm @ 176]
ExdiIpc!DllUnregisterServer+0x18a1d
ExdiIpc!DllUnregisterServer+0xff58
ucrtbase!thread_start<unsigned int (__cdecl*)(void *),1>+0x42
KERNEL32!BaseThreadInitThunk+0x14
ntdll!RtlUserThreadStart+0x21
```

进程关系图


OpenIPC_x64.exe
RPC
EXDIIPC服务进程
DllHost.exe
EXDI
WinDBG/NanoCode

进程关系



dllhost.exe		4,796 K	16,848 K	20544	COM Surrogate	Microsoft Corporation
conhost.exe	< 0.01	8,580 K	20,608 K	18480	Console Window Host	Microsoft Corporation
OpenIPC_x64.exe	1.51	69,852 K	92,192 K	8044	OpenIPC Debug Library (rev ...	Intel Corporation

- 父进程是Dcom Server Launcher 服务


工作原理
两种前端
调试NT内核
调试BootMgr
高级命令

WinDBG前端

- set IPC_PATH=C:\IntelSWTools\system_studio_2020\tools\OpenIPC_1.2011.4508.100\\Bin

- "C:\Wd10x64\windbg" -kx exdi:clsid={66664CB8-5320-4686-837D-7D17679728E6},Kd=Guess

搜索NT

- 64位线性地址，48位地址空间
- 256TB地址空间，上下各128TB

- 没有KD主动报告了
- 要搜！

启发式搜索

- 暴力搜索128TB太愚蠢了

找到NT内核

Kernel Debugger connection established

Found Module Name ntkrnlmp

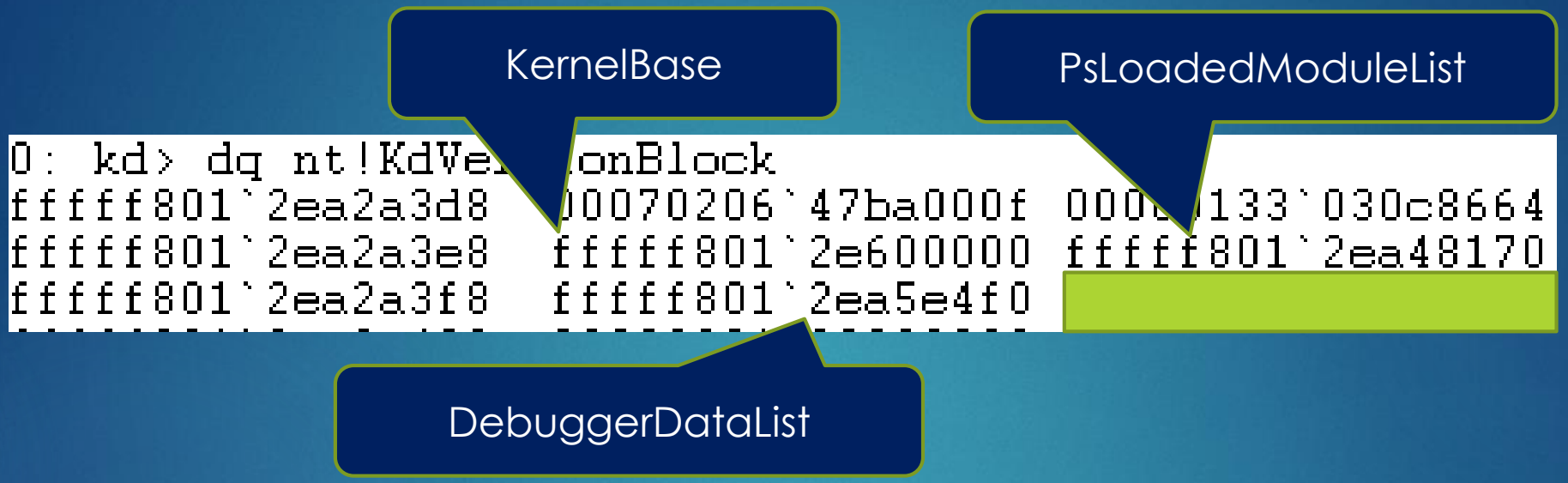
Found KdVersionBlock at 0xfffff8012ea2a3d8
Found target VersionBlock at 0xfffff8012ea2a3d8
Connected to Windows 10 18362 x64 target at (Sat Jun 6 17:54:13.684 2020 (UTC + 8:00)), ptr64 TRUE

寻找KdVersionBlock

- NT的版本很多
- 布局和细节差别很大
- 一个字节的偏差就可能访问空指针

```
typedef struct _DBGKD_GET_VERSION64 {
    USHORT  MajorVersion;
    USHORT  MinorVersion;
    UCHAR   ProtocolVersion;
    UCHAR   KdSecondaryVersion; // Cannot be 'A' for compat with dump header
    USHORT  Flags;
    USHORT  MachineType;
    //
    // Protocol command support descriptions.
    // These allow the debugger to automatically
    // adapt to different levels of command support
    // in different kernels.
    //

    // One beyond highest packet type understood, zero based.
    UCHAR   MaxPacketType;
    // One beyond highest state change understood, zero based.
    UCHAR   MaxStateChange;
    // One beyond highest state manipulate message understood, zero based.
    UCHAR   MaxManipulate;

    // Kind of execution environment the kernel is running in,
    // such as a real machine or a simulator.  Written back
    // by the simulation if one exists.
    UCHAR   Simulation;

    USHORT  Unused[1];

    ULONG64 KernBase;
    ULONG64 PsLoadedModuleList;

    //
    // Components may register a debug data block for use by
    // debugger extensions.  This is the address of the list head.
    //
    // There will always be an entry for the debugger.
    //
    ULONG64 DebuggerDataList;
} DBGKD_GET_VERSION64, *PDBGKD_GET_VERSION64;
```

宝贵的40个字节



- 内核基地址
- 模块列表
- 调试数据列表

```
//
// This structure is used by the debugger for all targets
// It is the same size as DBGKD_DATA_HEADER on all systems
//
typedef struct _DBGKD_DEBUG_DATA_HEADER64 {

    //
    // Link to other blocks
    //

    LIST_ENTRY64 List;

    //
    // This is a unique tag to identify the owner of the block.
    // If your component only uses one pool tag, use it for this, too.
    //

    ULONG          OwnerTag;

    //
    // This must be initialized to the size of the data block,
    // including this structure.
    //

    ULONG          Size;

} DBGKD_DEBUG_DATA_HEADER64, *PDBGKD_DEBUG_DATA_HEADER64;
```

```
typedef struct _KDDEBUGGER_DATA64 {

    DBGKD_DEBUG_DATA_HEADER64 Header;

    //
    // Base address of kernel image
    //

    ULONG64   KernBase;

    //
    // DbgBreakPointWithStatus is a function which takes an argument
    // and hits a breakpoint.  This field contains the address of the
    // breakpoint instruction.  When the debugger sees a breakpoint
    // at this address, it may retrieve the argument from the first
    // argument register, or on x86 the eax register.
    //

    ULONG64   BreakpointWithStatus;       // address of breakpoint

    //
    // Address of the saved context record during a bugcheck
    //
    // N.B. This is an automatic in KeBugcheckEx's frame, and
    // is only valid after a bugcheck.
    //

    ULONG64   SavedContext;

    //
    // help for walking stacks with user callbacks:
    //

    //
    // The address of the thread structure is provided in the
    // WAIT_STATE_CHANGE packet.  This is the offset from the base of
    // the thread structure to the pointer to the kernel stack frame
    // for the currently active usermode callback.
```

调试器数据的开头部分

• • •

```
//
// Addresses of various kernel data structures and lists
// that are of interest to the kernel debugger.
//

ULONG64   PsLoadedModuleList;
ULONG64   PsActiveProcessHead;
ULONG64   PspCidTable;

ULONG64   ExpSystemResourcesList;
ULONG64   ExpPagedPoolDescriptor;
ULONG64   ExpNumberOfPagedPools;

ULONG64   KeTimeIncrement;
ULONG64   KeBugCheckCallbackListHead;
ULONG64   KiBugcheckData;

ULONG64   IopErrorLogListHead;

ULONG64   ObpRootDirectoryObject;
ULONG64   ObpTypeObjectType;

ULONG64   MmSystemCacheStart;
ULONG64   MmSystemCacheEnd;
ULONG64   MmSystemCacheWs;

ULONG64   MmPfnDatabase;
ULONG64   MmSystemPtesStart;
ULONG64   MmSystemPtesEnd;
ULONG64   MmSubsectionBase;
ULONG64   MmNumberOfPagingFiles;

ULONG64   MmLowestPhysicalPage;
ULONG64   MmHighestPhysicalPage;
ULONG64   MmNumberOfPhysicalPages;
```

中间部分
关键结构体的地址

```
// Server 2003 addition

ULONG64   MmVirtualTranslationBase;

USHORT    OffsetKThreadNextProcessor;
USHORT    OffsetKThreadTeb;
USHORT    OffsetKThreadKernelStack;
USHORT    OffsetKThreadInitialStack;

USHORT    OffsetKThreadApcProcess;
USHORT    OffsetKThreadState;
USHORT    OffsetKThreadBStore;
USHORT    OffsetKThreadBStoreLimit;

USHORT    SizeEProcess;
USHORT    OffsetEprocessPeb;
USHORT    OffsetEprocessParentCID;
USHORT    OffsetEprocessDirectoryTableBase;

USHORT    SizePrcb;
USHORT    OffsetPrcbDpcRoutine;
USHORT    OffsetPrcbCurrentThread;
USHORT    OffsetPrcbMhz;

USHORT    OffsetPrcbCpuType;
USHORT    OffsetPrcbVendorString;
USHORT    OffsetPrcbProcStateContext;
USHORT    OffsetPrcbNumber;

USHORT    SizeEThread;

ULONG64   KdPrintCircularBufferPtr;
ULONG64   KdPrintBufferSize;

ULONG64   KeLoaderBlock;

USHORT    SizePcr;
USHORT    OffsetPcrSelfPcr;
```

关键字段的偏移位置

调试器数据

```
0: kd> dq fffff801`2ea5e4f0
fffff801`2ea5e4f0  fffff801`2ea265e0 fffff801`2ea265e0
fffff801`2ea5e500  fffff801`2ea5e500 fffff801`2ea5e500
```

```
0: kd> dq fffff801`2ea265e0
fffff801`2ea265e0  2635f064`714b4df3 2635f064`714b4df3
fffff801`2ea265f0  be2a4f9b`8af14385 2635f064`7e4501a3
fffff801`2ea26600  2635f064`704f1063 c62a7f9b`8e4507a1
fffff801`2ea26610  c62a7f8b`8e4507a1 2635f064`78409063
fffff801`2ea26620  c62a7f9b`8e4507a1 2635f064`794d1de3
fffff801`2ea26630  2635f064`784dbd93 2635f064`7d415cd3
fffff801`2ea26640  2635f064`70445d93 2635f064`724d2cd3
fffff801`2ea26650  2635f064`7bc60cd3 2635f064`70413cd3
```

?

- 加了密了

原来如此啊

****

Unable to read debugger data block header

****

****

Unable to read debugger data block header

****

这个问题有办法解决么？

NANO DEBUGGER: ...

- Open Executable
- Attach to a process
- Kernel Debugging
- Open Crash Dump

Welcome | Nano Home | Nano Debugger

File View

```
PROCESS ffffa70426bf84c0
SessionId: 0 Cid: 1858 Peb: ea64788000 ParentCid: 02e8
DirBase: 0828b002 ObjectTable: ffffdc834aea3100 HandleCount: 230.
Image: svchost.exe

PROCESS ffffa704240f4080
SessionId: 0 Cid: 0d68 Peb: d8e6f3c000 ParentCid: 02e8
DirBase: 16996e002 ObjectTable: ffffdc8347ec3080 HandleCount: 113.
Image: svchost.exe

PROCESS ffffa704262ce080
SessionId: 0 Cid: 1f08 Peb: 951cb8a000 ParentCid: 02e8
DirBase: 420a0002 ObjectTable: ffffdc8348b9c180 HandleCount: 160.
Image: svchost.exe

PROCESS ffffa70426ca30c0
SessionId: 1 Cid: 17c0 Peb: 97e6c74000 ParentCid: 03d8
DirBase: 6d638002 ObjectTable: ffffdc834975f100 HandleCount: 269.
Image: RuntimeBroker.exe

g
Switched to processor 1, its device id is 0x1001
Switched to processor 2, its device id is 0x1002
Switched to processor 3, its device id is 0x1003
```

BUSY Running... 0% 100%

TERMINAL DEBUG CONSOLE OUTPUT PROBLEMS

Nano Code

```
18:53:13#x8664:Segment ES: base 0 limit ffffffff flags c0f3
18:53:13#x8664:Segment FS: base 0 limit fc00 flags 40f3
18:53:13#x8664:Segment GS: base ffff930060480000 limit ffffffff flags c0f3
18:53:13#x8664:Segment SS: base 0 limit 0 flags 4093
18:53:13#EVCB:ChangeEngineState: CES_EXECUTION_STATUS[0x10], 0x1
18:53:13#JTAG:Go/Step (0) command was sent to JTAG chip with handle 5709386
18:53:13#EXDI:StartNotifyingRunChg3
18:53:13#JTGE:JtagRunControlEvent:type=65 deviceid=4096 dc=1, name= type= subtype= @0
18:53:13#JTGE:JtagRunControlEvent:type=65 deviceid=4097 dc=1, name= type= subtype= @0
18:53:13#JTGE:JtagRunControlEvent:type=65 deviceid=4098 dc=1, name= type= subtype= @0
18:53:13#JTGE:JtagRunControlEvent:type=65 deviceid=4099 dc=1, name= type= subtype= @0
18:53:14#EXDI:run status 0x0, 0xfffff801485f4332, 0xe0000002
```

变化

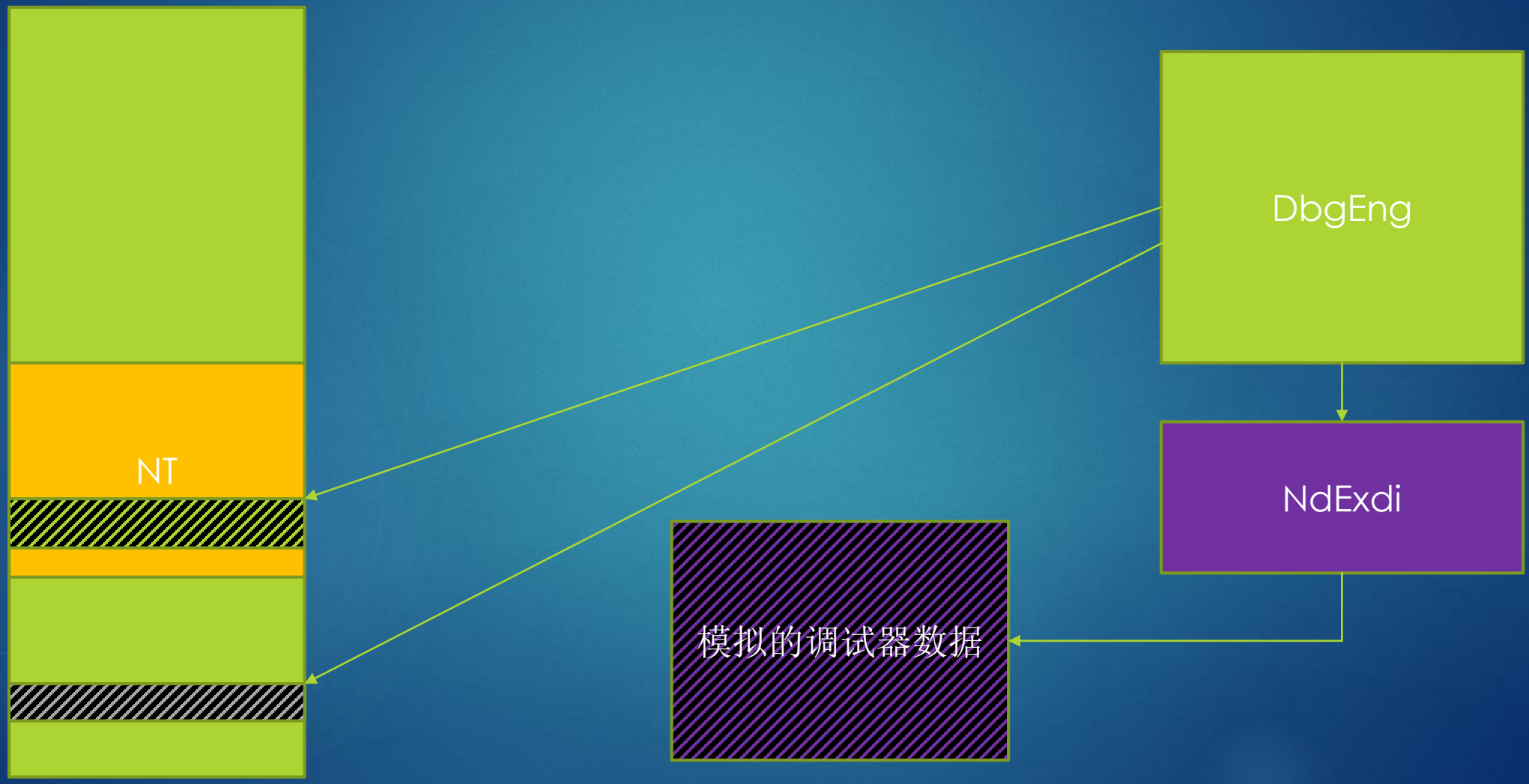
- 使用Nano Debugger调试时看不见那个错误信息了

为什么

```
dq nt!KdVersionBlock
fffff801`2ea2a3d8 00070206`47ba000f 00000133`030c8664
fffff801`2ea2a3e8 fffff801`2e600000 fffff801`2ea48170
fffff801`2ea2a3f8 fffd0000`00000008 00000001`00000fa0
```

变了个地址

李代桃僵



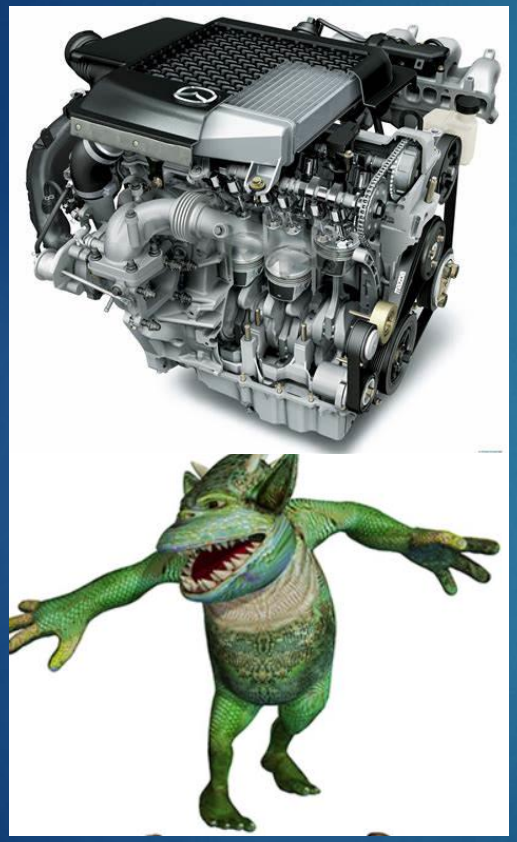
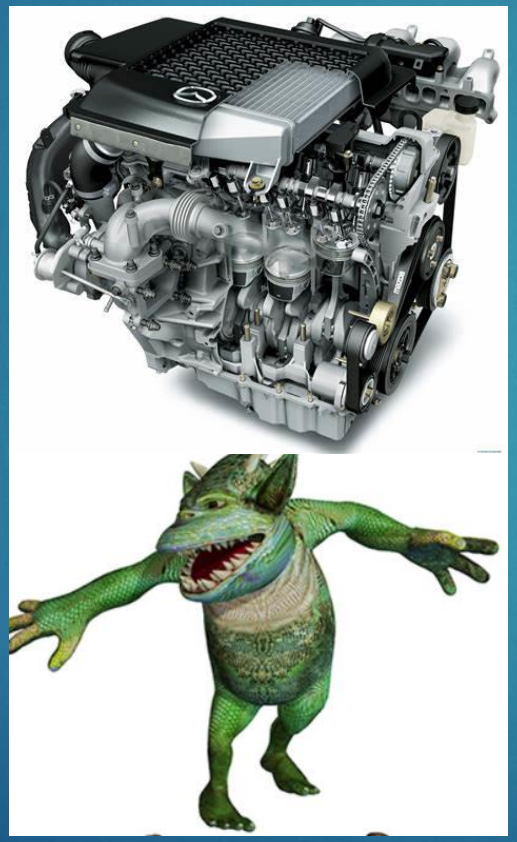
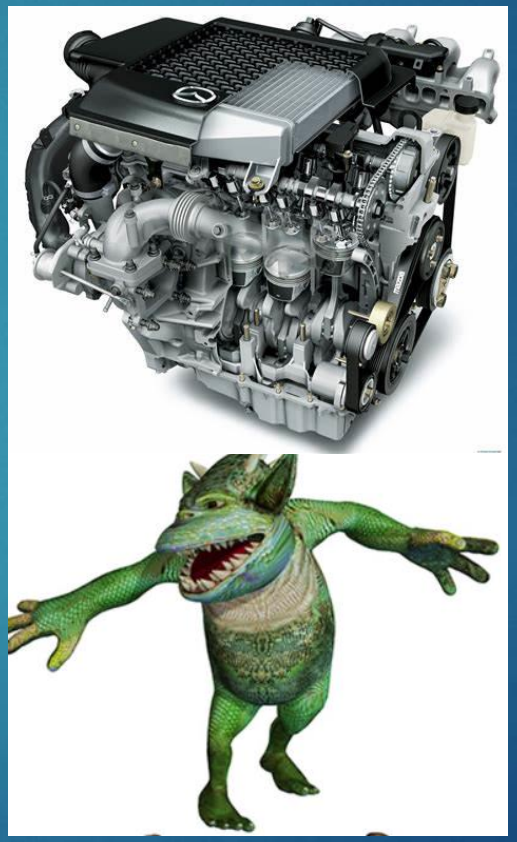
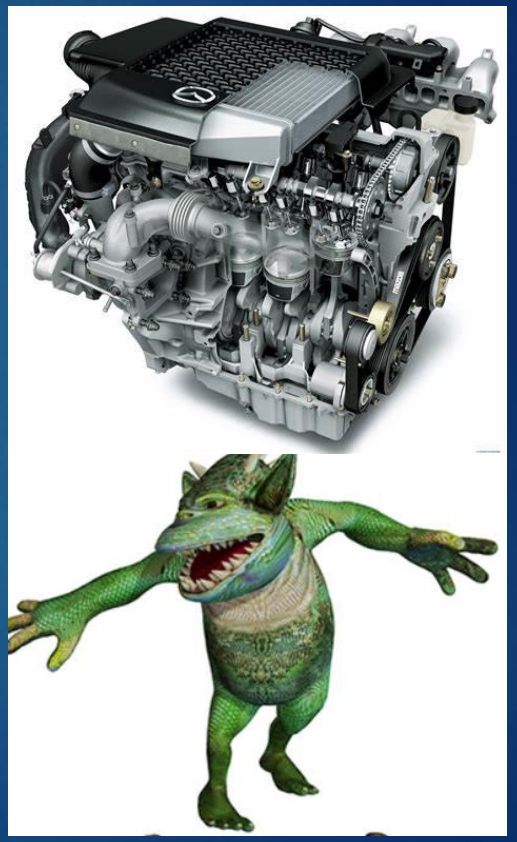

工作原理
两种前端
调试NT内核
调试BootMgr
高级命令

CPU管理




系统的动力来源，“生命”之源

每个都是神兽

面向对象

KPCR

KPCR

KPCR

KPCR

特殊的访问方式

```
nt!KeGetCurrentThread:
fffff801`2e701630 65488b042588010000 mov rax,qword
ptr gs:[188h]
fffff801`2e701639 c3 ret
```

KPCR

!pcr KPCR for Processor 0 at fffff8012c6c7000:
Major 1 Minor 1
NtTib.ExceptionList: fffff80134290fb0
NtTib.StackBase: fffff8013428f000
NtTib.StackLimit: 0000000000000000
NtTib.SubSystemTib: fffff8012c6c7000
NtTib.Version: 000000002c6c7180
NtTib.UserPointer: fffff8012c6c7870
NtTib.SelfTib: 0000006dd3785000

使用dt命令

- dt _KPCR fffff8012c6c7000
- dt _KPCR fffff8012c6c7000 -r

```
Dumping IDT: fffff8013428e000

00: fffff8012e951100 nt!KiDivideErrorFaultShadow
01: fffff8012e951180 nt!KiDebugTrapOrFaultShadow Stack = 0xFFFFF801342929D0
02: fffff8012e951200 nt!KiNmiInterruptShadow Stack = 0xFFFFF801342927D0
03: fffff8012e951280 nt!KiBreakpointTrapShadow
04: fffff8012e951300 nt!KiOverflowTrapShadow
05: fffff8012e951380 nt!KiBoundFaultShadow
06: fffff8012e951400 nt!KiInvalidOpcodeFaultShadow
07: fffff8012e951480 nt!KiNpxNotAvailableFaultShadow
08: fffff8012e951500 nt!KiDoubleFaultAbortShadow Stack = 0xFFFFF801342923D0
09: fffff8012e951580 nt!KiNpxSegmentOverrunAbortShadow
0a: fffff8012e951600 nt!KiInvalidTssFaultShadow
0b: fffff8012e951680 nt!KiSegmentNotPresentFaultShadow
0c: fffff8012e951700 nt!KiStackFaultShadow
0d: fffff8012e951780 nt!KiGeneralProtectionFaultShadow
0e: fffff8012e951800 nt!KiPageFaultShadow
10: fffff8012e951880 nt!KiFloatingErrorFaultShadow
11: fffff8012e951900 nt!KiAlignmentFaultShadow
12: fffff8012e951980 nt!KiMcheckAbortShadow Stack = 0xFFFFF801342925D0
13: fffff8012e951a80 nt!KiXmmExceptionShadow
14: fffff8012e951b00 nt!KiVirtualizationExceptionShadow
15: fffff8012e951b80 nt!KiControlProtectionFaultShadow
1f: fffff8012e951c00 nt!KiApcInterruptShadow
20: fffff8012e951c80 nt!KiSwInterruptShadow
29: fffff8012e951d00 nt!KiRaiseSecurityCheckFailureShadow
2c: fffff8012e951d80 nt!KiRaiseAssertionShadow
2d: fffff8012e951e00 nt!KiDebugServiceTrapShadow
2f: fffff8012e951f00 nt!KiDpcInterruptShadow
30: fffff8012e951f80 nt!KiHvInterruptShadow
31: fffff8012e952000 nt!KiVmbusInterrupt0Shadow
```

IDT

- 软硬件直接的关键纽带
- 最频繁的交接场地

硬件资源分配详情

```
DEVNODE ffffa7041fab9c10 (ACPI\PNP0A08\0)
Memory Arbiter "PCI Memory (b=0)" at ffffa7041f0d8bd0
Allocated ranges:
0000000000000000 - 000000000009ffff 00000000 <Not on bus>
00000000000a0000 - 00000000000bffff S ffffa7041facd060 (igfx)
00000000000c0000 - 00000008ffffffff 00000000 <Not on bus>
00000000c0000000 - 00000000cfffffff ffffa7041facd060 (igfx)
00000000d0000000 - 00000000d00fffff ffffa7041fa74060 (pci)
00000000de000000 - 00000000deffffff ffffa7041facd060 (igfx)
00000000df200000 - 00000000df2fffff ffffa7041fa74060 (pci)
00000000df310000 - 00000000df31ffff ffffa7041fad0060 (USBXHCI)
00000000df328000 - 00000000df329fff ffffa7041fa73060 (storahci)
00000000df32a000 - 00000000df32a0ff B ffffa7041fa1f060
00000000df32b000 - 00000000df32b7ff ffffa7041fa73060 (storahci)
00000000df32c000 - 00000000df32c0ff ffffa7041fa73060 (storahci)
00000000df32e000 - 00000000df32efff B ffffa7041fad2060
00000000dfc00000 - 00000000dfdfffff ffffa7041fad1060 (ufxsynopsys)
00000000dffe0000 - 00000000dfffffff BA ffffa7041facee00
```

- !arbiter

设备树

```
DevNode 0xffffa7041fa76670 for PDO 0xffffa7041fa6e530
InstancePath is "ACPI\PNP0C02\5"
State = DeviceNodeInitialized (0x302)
Previous State = DeviceNodeUninitialized (0x301)
DevNode 0xffffa7041fa249e0 for PDO 0xffffa7041f9199f0
InstancePath is "ACPI\GenuineIntel_-_Intel64_Family_6_Model_78_-_Intel(R)_Pentium(R)_CPU_4405U_@_2.10GHz\_1"
ServiceName is "intelppm"
State = DeviceNodeStarted (0x308)
Previous State = DeviceNodeEnumerateCompletion (0x30d)
DevNode 0xffffa7041fa949e0 for PDO 0xffffa7041f0cad50
InstancePath is "ACPI\GenuineIntel_-_Intel64_Family_6_Model_78_-_Intel(R)_Pentium(R)_CPU_4405U_@_2.10GHz\_2"
ServiceName is "intelppm"
State = DeviceNodeStarted (0x308)
Previous State = DeviceNodeEnumerateCompletion (0x30d)
DevNode 0xffffa7041faca9e0 for PDO 0xffffa7041f9b4cf0
InstancePath is "ACPI\GenuineIntel_-_Intel64_Family_6_Model_78_-_Intel(R)_Pentium(R)_CPU_4405U_@_2.10GHz\_3"
ServiceName is "intelppm"
State = DeviceNodeStarted (0x308)
Previous State = DeviceNodeEnumerateCompletion (0x30d)
DevNode 0xffffa7041fa839e0 for PDO 0xffffa7041f922cd0
InstancePath is "ACPI\GenuineIntel_-_Intel64_Family_6_Model_78_-_Intel(R)_Pentium(R)_CPU_4405U_@_2.10GHz\_4"
ServiceName is "intelppm"
```

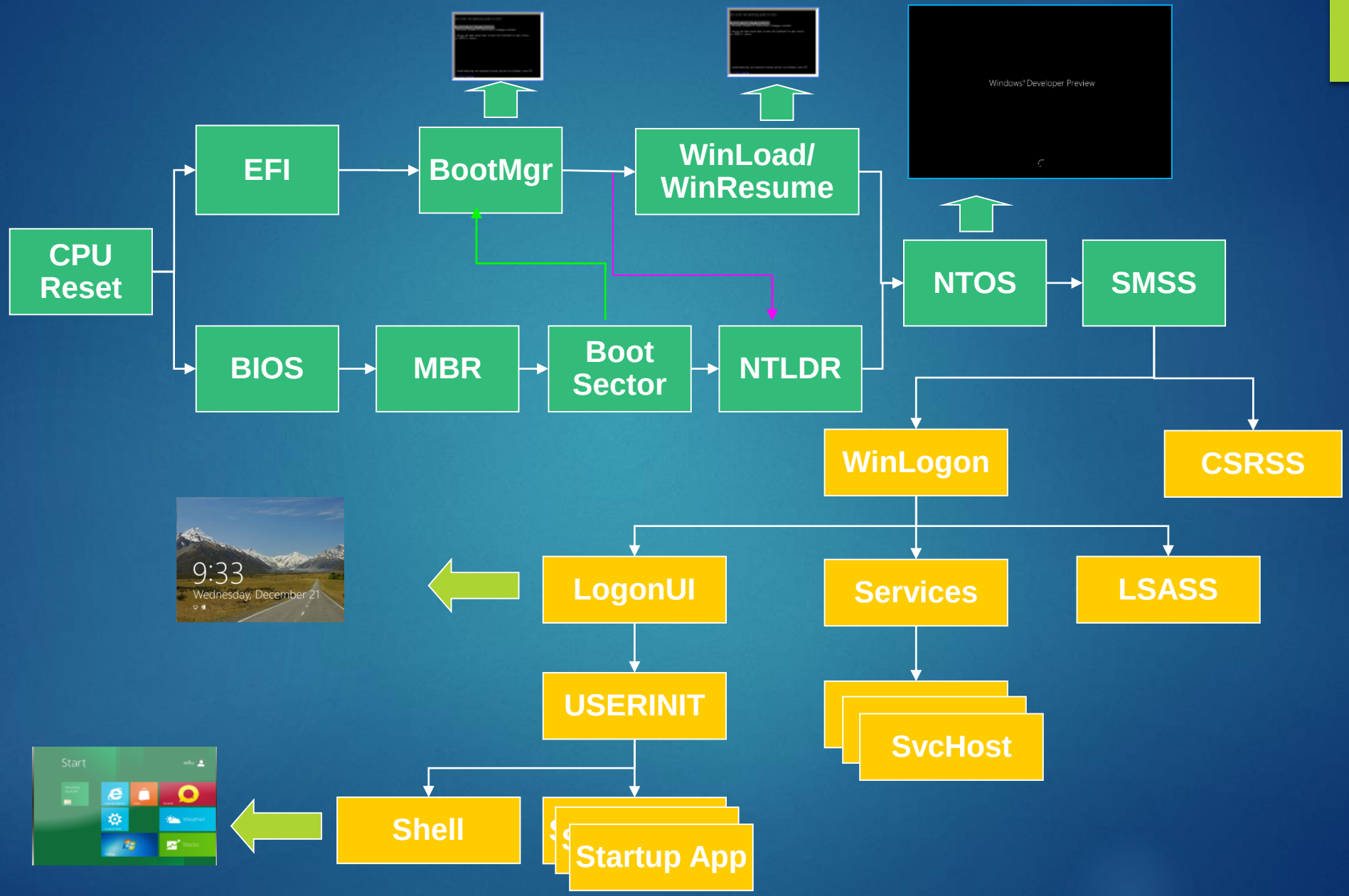
- !devnode 0 1

设备栈

```
[ndb]!devstack 0xffffa7041fad0060
!devstack 0xffffa7041fad0060
!DevObj !DrvObj !DevExt ObjectName
ffffa7041fb878f0 \Driver\USBXHCI ffffa7041fbe0ba0 USBFDO-0
ffffa7041fa20c20 \Driver\ACPI ffffa7041f9cf880
> ffffa7041fad0060 \Driver\pci ffffa7041fad01b0 NTPNP_PCI0003
!DevNode ffffa7041f922340 :
DeviceInst is "PCI\VEN_8086&DEV_9D2F&SUBSYS_72708086&REV_21\3&11583659&0&A0"
ServiceName is "USBXHCI"
```

- WDM
- NT中的经典
- 使用C语言实现面向对象


工作原理
两种前端
调试NT内核
调试BootMgr
高级命令

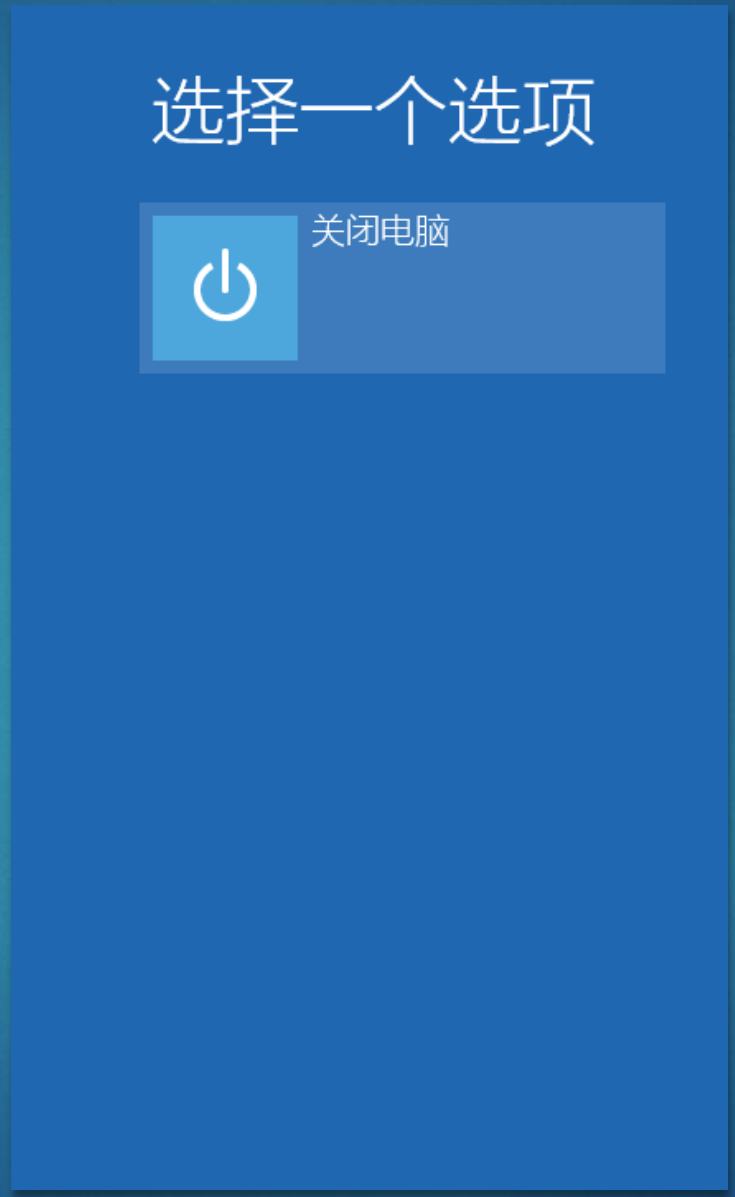

EFI
BootMgr
WinLoad/
WinResume
Windows® Developer Preview
CPU
Reset
NTOS
SMSS
BIOS
MBR
Boot
Sector
NTLDR
WinLogon
CSRSS
9:33
Wednesday, December 21
LogonUI
Services
LSASS
USERINIT
SvcHost
Start
Shell
Startup App

BOOTIM.exe

- Application
  - bcdedit /bootsequence {xxxxx}
- “Worst change in win8 ☹”
- Example: Win8 + Win7 duo system, and want start Win7
  - Start win8→bootim.exe→set temporary boot entry→reboot→let bootmgr load win7
- Bcdedit /set bootmenupolicy legacy
- bcdedit /set bootmenupolicy standard

Same in w10

- 可以在启动后运行起来
- Alt + Tab切换回其它窗口



克隆的调试设施

- KD
- KdVersionBlock

```
x bootmgfw!*versionb*
00000000`8560e220 bootmgfw!BdVersionBlock = <no type information>
```

```
x bootmgfw!*kd*
00000000`85607204 bootmgfw!KdNetUsb3ControllerReset = <no type information>
00000000`85607490 bootmgfw!KdNetRestartController = <no type information>
00000000`85607400 bootmgfw!KdNetDhcpDiscoverResponseTime = <no type information>
00000000`8560722c bootmgfw!KdNetUsb3InvalidTransferEventPointers = <no type information
00000000`855c3040 bootmgfw!KdHvDetectHypervisor (<no parameter info>)
00000000`8560e5dc bootmgfw!KdNetErrorStatus = <no type information>
00000000`85607228 bootmgfw!KdNetUsb3ReleaseRxPacketInvalidParameter = <no type informat
00000000`85607470 bootmgfw!KdNetRxPacketTooSmallForIpv6 = <no type information>
00000000`8560e5c0 bootmgfw!KdTargetIP = <no type information>
00000000`856073f8 bootmgfw!KdNetInitialConnectTime = <no type information>
00000000`85607494 bootmgfw!KdNetSentPingPacket = <no type information>
00000000`855bc0cc bootmgfw!KdUsbpLocateDebugPortRegisters (<no parameter info>)
00000000`8560749c bootmgfw!KdNetDhcpLeaseValid = <no type information>
00000000`85607138 bootmgfw!KdHvDetectionComplete = <no type information>
00000000`8560738c bootmgfw!KdNetKdReceivePacketBadPacketSize = <no type information>
00000000`855c3130 bootmgfw!KdHvConnectHypervisor (<no parameter info>)
00000000`85607440 bootmgfw!KdNetWaitForRxPacketStalls = <no type information>
00000000`85607464 bootmgfw!KdNetRxPacketsFailed = <no type information>
00000000`855b6684 bootmgfw!KdUsbReceivePacket (<no parameter info>)
00000000`855b61cc bootmgfw!KdVmInitializeController (<no parameter info>)
00000000`855bb110 bootmgfw!KdUsb3pReadDebuggerPacket (<no parameter info>)
00000000`85607380 bootmgfw!KdNetBailPrintString = <no type information>
00000000`85607240 bootmgfw!KdNetUsb3EventsSkipped = <no type information>
00000000`85529398 bootmgfw!PltpCheckDeviceExistence (<no parameter info>)
00000000`85607220 bootmgfw!KdNetUsb3ProcessEventsReset = <no type information>
00000000`856074d0 bootmgfw!KdNetHandleIcmpv6Multicast = <no type information>
00000000`856074b8 bootmgfw!KdNetGratuitousArpsSent = <no type information>
```


工作原理
两种前端
调试NT内核
调试BootMgr
高级命令

扩展命令

- !ds
- !sxe *
- !sxd *
- !sxe Hlt

千头万绪，终归一理

宋·朱熹 (1130-1200)

所谓致知在格物者，言欲致吾之知，在**即物**而穷其理也。盖人心之灵莫不有知，而天下之物莫不有理，惟于理有未穷，故其知有不尽也。是以《大学》始教，必使学者即凡天下之物，莫不因其已知之理而益穷之，以求至乎其极。至于用力之久，而一旦豁然**贯通**焉，则众物之表里精粗无不到，而吾心之全体大用无不明矣。此谓物格，此谓知之至也。

问答

http://www.nanocode.cn

http://advdbg.org/gdk