

# CS490 Windows Internals Labs

Sep 13<sup>th</sup>, 2013

## 1. Viewing the Process Tree

### Tlist

Usually, you can retrieve most of information of processes from task manager, except the parent process ID. In this lab, we are going to use Tlist.exe tool to get parent ID of a process. Tlist.exe can be found in your Debugging Tools for Windows installation directory. To show the process tree, use /t switch. The format to call Tlist.exe is: tlist /t

Here's an output example of Tlist.exe.

```
C:\Program Files\Debugging Tools for Windows (x64)>tlist /t
System Process (0)
System (4)
  smss.exe (244)
  csrss.exe (364)
  wininit.exe (440)
  services.exe (512)
    svchost.exe (668)
      dllhost.exe (5212)
        companionuser.exe (8744)
        wlcomm.exe (8484) MSNUnnamedWindow
        WmiPrvSE.exe (3464)
        dllhost.exe (3176) OleMainThreadWndName
      svchost.exe (752)
      atiesrxx.exe (800)
        atieclxx.exe (1208) AMD EEU Client
      svchost.exe (888)
        audiodg.exe (7208)
      svchost.exe (928)
        dwm.exe (2068) DWM Notification Window
      svchost.exe (956)
      svchost.exe (396)
      ZhuDongFangYu.exe (376)
      svchost.exe (1064)
      svchost.exe (1236)
      AsLdrSrv.exe (1344)
        wcourier.exe (2408) Wireless Console
        HControl.exe (2332) HControl
        ATKOSD.exe (3080) ATKOSD
        WDC.exe (3104) Windows Device Control
```

Here, you can see that if a process doesn't have a parent, it is left-justified. Windows only maintain the parent process ID, so that even if these processes have grandparents, Tlist.exe cannot show them to you.

To prove that, follow these steps:

1. Open a Command Prompt Window.
2. Type **start cmd** to start a new Command Prompt Window.
3. In the second command prompt, type **mspaint** to run Microsoft Paint.
4. Use **Tlist.exe** to check the tree of current processes, just as the following picture:

```
cmd.exe (264) 管理员: C:\Windows\system32\cmd.exe
cmd.exe (7108) 管理员: C:\Windows\system32\cmd.exe
mspaint.exe (848) 无标题 - 画图
```

5. Switch to the second command prompt, type **exit**. (Windows Paint remains there)
6. Check the process tree by **Tlist.exe**, and you can see the Microsoft Paint is left-justified now.

```
mspaint.exe (848) 无标题 - 画图
C:\Program Files\Debugging Tools for Windows (x64)>
```

7. At this time, bring up **Task Manager**, click the **Applications** tab, right-click on the Command Prompt task, and select **Go To Process**.
8. Right-click on the **cmd.exe** highlighted, and select **End Process Tree** to end the process tree.

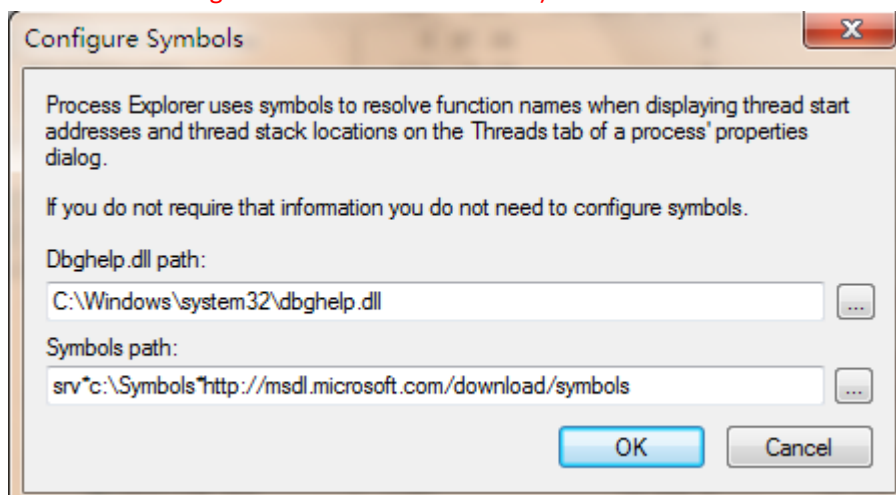
Now you can still see Microsoft paint, but the command prompt disappeared. That's because Windows have not link between a grandparent or a grandchild.

## Process Explorer

In this lab, instead of using Tlist.exe, you can also use Process Explorer in sysinternals. Process Explorer can return much more information of processes to you than Tlist.

1. Run **procxp.exe** from the sysinternals package.
2. The first time you open Process Explorer, you may need to configure the symbol path. Set the symbol path as follows (If you have downloaded a local package of symbols, set the path to your symbol directory).

Microsoft symbol server: <http://msdl.microsoft.com/download/symbols> (OLD please try to find new link and get familiar with the TechNet)



3. Click **OK** and you can see the main window of Process Explorer. Just enjoy it.

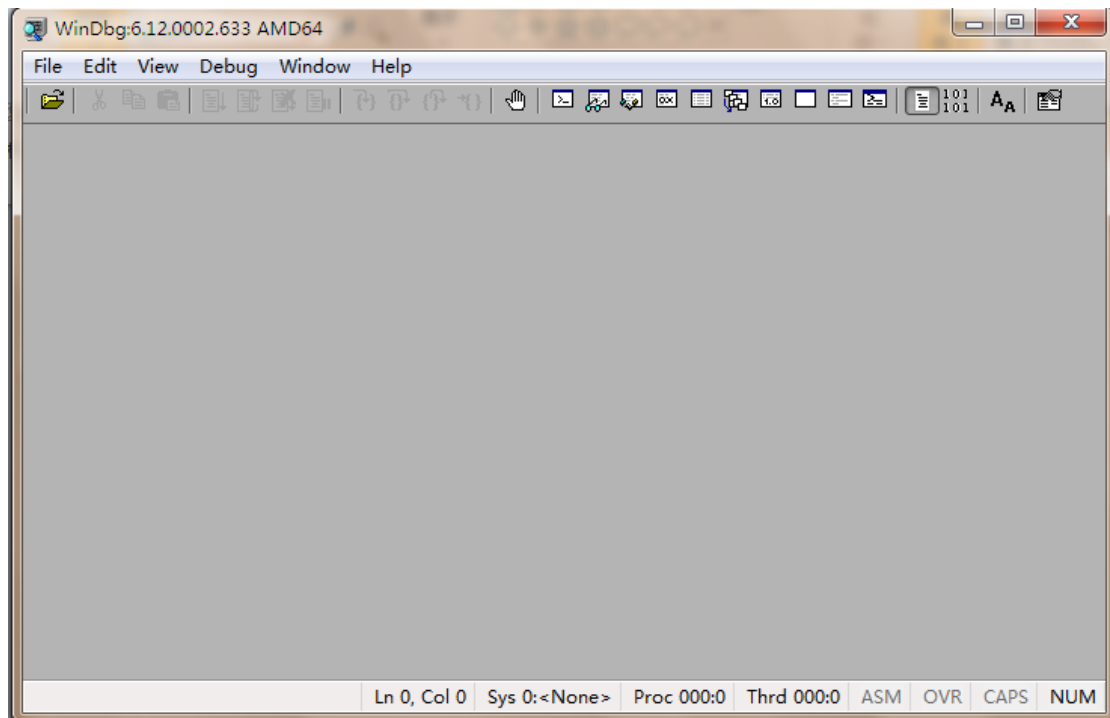
Process	PID	CPU	Private Bytes	Working Set	Description
System Idle Process	0	86.75		24 K	
Interrupts	n/a				K Hardware Inter
DPCs	n/a	0.76			K Deferred Proce
System	4		112 K	368 K	
smss.exe	244		456 K	1,124 K	Windows 会话管
csrss.exe	364		2,988 K	6,820 K	Client Server
wininit.exe	440		1,520 K	4,920 K	Windows 启动应
services.exe	512		6,784 K	12,244 K	服务和控制器应用
svchost.exe	668		4,464 K	9,876 K	Windows 服务主
dllhost.exe	5212		2,444 K	6,884 K	COM Surrogate
svchost.exe	752		5,592 K	10,088 K	Windows 服务主
atiesrxx.exe	800		1,416 K	4,168 K	AMD External E
atieclxx.exe	1208		2,000 K	5,708 K	AMD External E
svchost.exe	888		21,096 K	24,504 K	Windows 服务主
audiodg.exe	7780		18,884 K	19,392 K	Windows 音频设
svchost.exe	928		147,060 K	147,432 K	Windows 服务主
dwm.exe	2068	3.80	37,216 K	42,312 K	桌面窗口管理器
svchost.exe	956		18,396 K	33,772 K	Windows 服务主
svchost.exe	396		10,324 K	18,984 K	Windows 服务主
ZhuDongFangYu.exe	376		3,616 K	8,696 K	360主动防御服务
svchost.exe	1064		26,044 K	30,800 K	Windows 服务主
svchost.exe	1236		18,064 K	22,456 K	Windows 服务主
AsLdrSrv.exe					

CPU Usage: 12.17% Commit Charge: 39.55% Processes: 77 Physical Usage: 72.07%

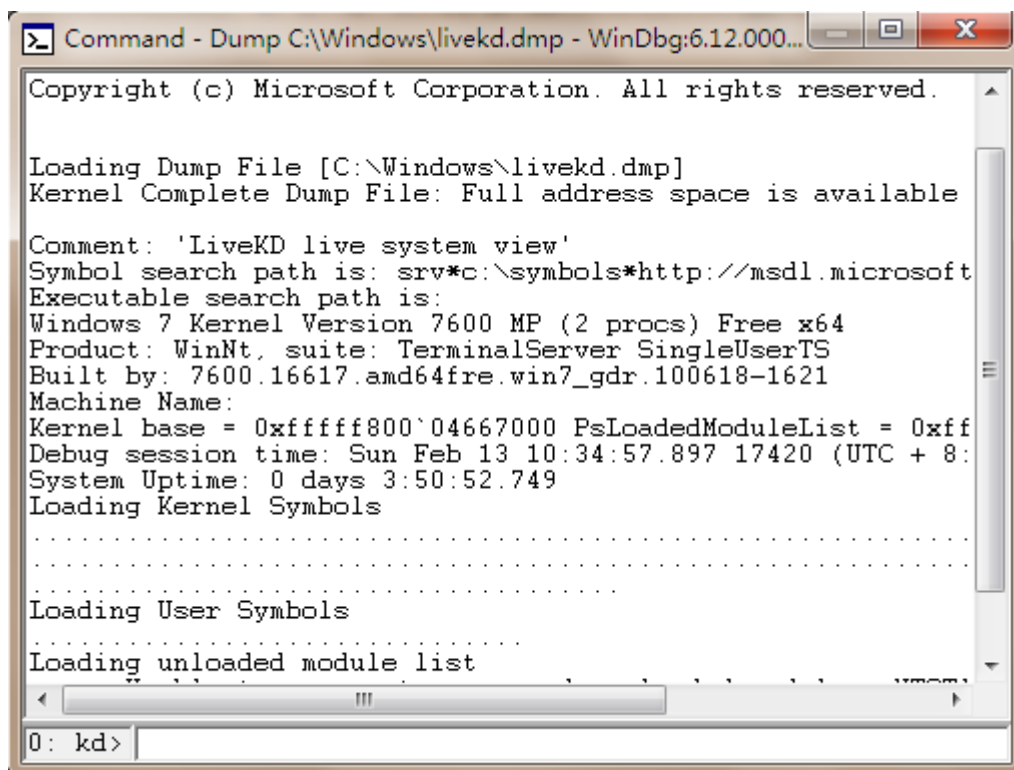
## 2. Kernel Debugging

Debugging Tools for Windows package contains many tools for debugging Windows. These tools can be used to debug user-mode as well as the kernel. Here you are going to do an experiment on kernel debugging by using Windbg, which is the GUI version of windows debugger.

1. Before using Windbg, please boot Windows in Debugging mode. Press **F8** before windows started, select **Debugging Mode** to boot the system.
2. Windbg.exe is in the directory of debugging tools for Windows. The first time you run it, you must configure the symbol path. (To configure symbol path, please read the guide.pdf on the course website). After that, you can see the following window:

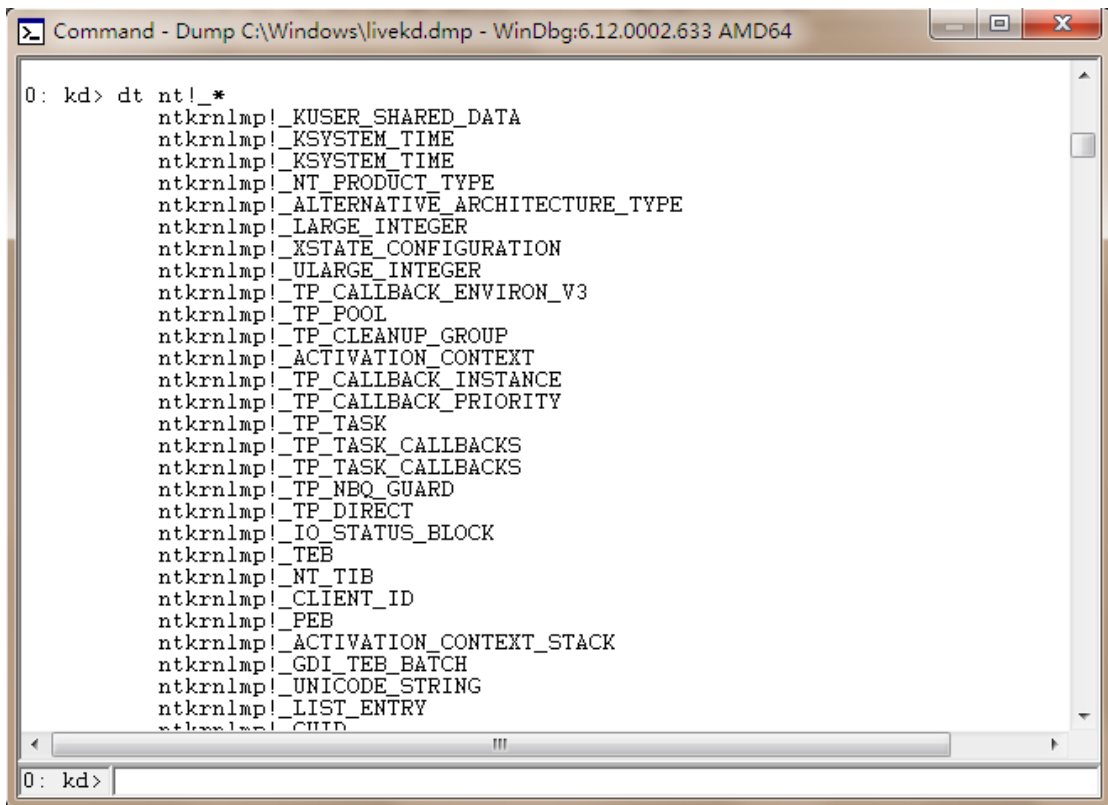


3. Click **"File"** on the menu bar, and select **"Kernel Debug..."**, in the Kernel Debugging dialog window, choose **"local"** and click **"OK"**, and **"Yes"** to save the information for workspace. The user interface of windbg is like this:



4. In the bottom of the UI, you can input debug commands. In this lab, we just try the display type command **"dt"**, to display the list of kernel structures whose type information is

included in the kernel symbols. A simple use of the “dt” command is `dt nt!_*`, which return all of the kernel structures to you.



```
Command - Dump C:\Windows\livekd.dmp - WinDbg:6.12.0002.633 AMD64
0: kd> dt nt!*
ntkrnlmp!_KUSER_SHARED_DATA
ntkrnlmp!_KSYSTEM_TIME
ntkrnlmp!_KSYSTEM_TIME
ntkrnlmp!_NT_PRODUCT_TYPE
ntkrnlmp!_ALTERNATIVE_ARCHITECTURE_TYPE
ntkrnlmp!_LARGE_INTEGER
ntkrnlmp!_XSTATE_CONFIGURATION
ntkrnlmp!_ULARGE_INTEGER
ntkrnlmp!_TP_CALLBACK_ENVIRON_V3
ntkrnlmp!_TP_POOL
ntkrnlmp!_TP_CLEANUP_GROUP
ntkrnlmp!_ACTIVATION_CONTEXT
ntkrnlmp!_TP_CALLBACK_INSTANCE
ntkrnlmp!_TP_CALLBACK_PRIORITY
ntkrnlmp!_TP_TASK
ntkrnlmp!_TP_TASK_CALLBACKS
ntkrnlmp!_TP_TASK_CALLBACKS
ntkrnlmp!_TP_NBQ_GUARD
ntkrnlmp!_TP_DIRECT
ntkrnlmp!_IO_STATUS_BLOCK
ntkrnlmp!_TEB
ntkrnlmp!_NT_TIB
ntkrnlmp!_CLIENT_ID
ntkrnlmp!_PEB
ntkrnlmp!_ACTIVATION_CONTEXT_STACK
ntkrnlmp!_GDI_TEB_BATCH
ntkrnlmp!_UNICODE_STRING
ntkrnlmp!_LIST_ENTRY
ntkrnlmp!_CURD
```

5. You can use “dt” to search kernel structures in many forms, such as `dt nt!*process*`. In this case the debugger will return structures that contain the term “process” to you.

```
0: kd> dt nt!*process*
ntkrnlmp!_PROCESSOR_NUMBER
ntkrnlmp!_KPROCESSOR_STATE
ntkrnlmp!_PROCESSOR_POWER_STATE
ntkrnlmp!_KPROCESS
ntkrnlmp!_EPROCESS
ntkrnlmp!_EPROCESS_QUOTA_BLOCK
ntkrnlmp!_SE_AUDIT_PROCESS_CREATION_INFO
ntkrnlmp!_ALPC_PROCESS_CONTEXT
ntkrnlmp!_PCW_PROCESSOR_INFO
ntkrnlmp!_PCW_PROCESSOR_INFO
ntkrnlmp!_RTL_USER_PROCESS_PARAMETERS
ntkrnlmp!_OBJECT_HEADER_PROCESS_INFO
ntkrnlmp!_PROCESSOR_CACHE_TYPE
ntkrnlmp!_KPROCESS_STATE
```

6. Notice that the “\*” here means more than 0 character, and you can also try “?”, which means more than 1 character.

```

0: kd> dt nt!*process
ntkrnlmp!_KPROCESS
ntkrnlmp!_EPROCESS
0: kd> dt nt!_?process*
ntkrnlmp!_KPROCESSOR_STATE
ntkrnlmp!_KPROCESS
ntkrnlmp!_EPROCESS
ntkrnlmp!_EPROCESS_QUOTA_BLOCK
ntkrnlmp!_KPROCESS_STATE
0: kd> dt nt!*process?

```

7. If you want to see more detail, try this: **dt nt!\_KPROCESS**, and you can see the inner structure of type KPROCESS.

```

0: kd> dt nt!_KPROCESS
+0x000 Header : _DISPATCHER_HEADER
+0x018 ProfileListHead : _LIST_ENTRY
+0x028 DirectoryTableBase : Uint8B
+0x030 ThreadListHead : _LIST_ENTRY
+0x040 ProcessLock : Uint8B
+0x048 Affinity : _KAFFINITY_EX
+0x070 ReadyListHead : _LIST_ENTRY
+0x080 SwapListEntry : _SINGLE_LIST_ENTRY
+0x088 ActiveProcessors : _KAFFINITY_EX
+0x0b0 AutoAlignment : Pos 0, 1 Bit
+0x0b0 DisableBoost : Pos 1, 1 Bit
+0x0b0 DisableQuantum : Pos 2, 1 Bit
+0x0b0 ActiveGroupsMask : Pos 3, 4 Bits
+0x0b0 ReservedFlags : Pos 7, 25 Bits
+0x0b0 ProcessFlags : Int4B
+0x0b4 BasePriority : Char
+0x0b5 QuantumReset : Char
+0x0b6 Visited : UChar
+0x0b7 Unused3 : UChar
+0x0b8 ThreadSeed : [4] Uint4B
+0x0c8 IdealNode : [4] Uint2B
+0x0d0 IdealGlobalNode : Uint2B
+0x0d2 Flags : _KEXECUTE_OPTIONS
+0x0d3 Unused1 : UChar
+0x0d4 Unused2 : Uint4B
+0x0d8 Unused4 : Uint4B
+0x0dc StackCount : _KSTACK_COUNT
+0x0e0 ProcessListEntry : _LIST_ENTRY
+0x0f0 CycleTime : Uint8B

```

8. To view the structure trees, use the **-r** option, like this: **dt nt!\_KPROCESS -r**.

```

0: kd> dt nt!_KPROCESS -r
+0x000 Header : _DISPATCHER_HEADER
+0x000 Type : UChar
+0x001 TimerControlFlags : UChar
+0x001 Absolute : Pos 0, 1 Bit
+0x001 Coalescable : Pos 1, 1 Bit
+0x001 KeepShifting : Pos 2, 1 Bit
+0x001 EncodedTolerableDelay : Pos 3, 5 Bits
+0x001 Abandoned : UChar
+0x001 Signalling : UChar
+0x002 ThreadControlFlags : UChar
+0x002 CpuThrottled : Pos 0, 1 Bit
+0x002 CycleProfiling : Pos 1, 1 Bit
+0x002 CounterProfiling : Pos 2, 1 Bit
+0x002 Reserved : Pos 3, 5 Bits
+0x002 Hand : UChar
+0x002 Size : UChar

```

For more commands, please check the **Debugging Help** for Windows debugging tools.