CS490 Windows Internals Labs

Sep 30th, 2013

Process Explorer Lab

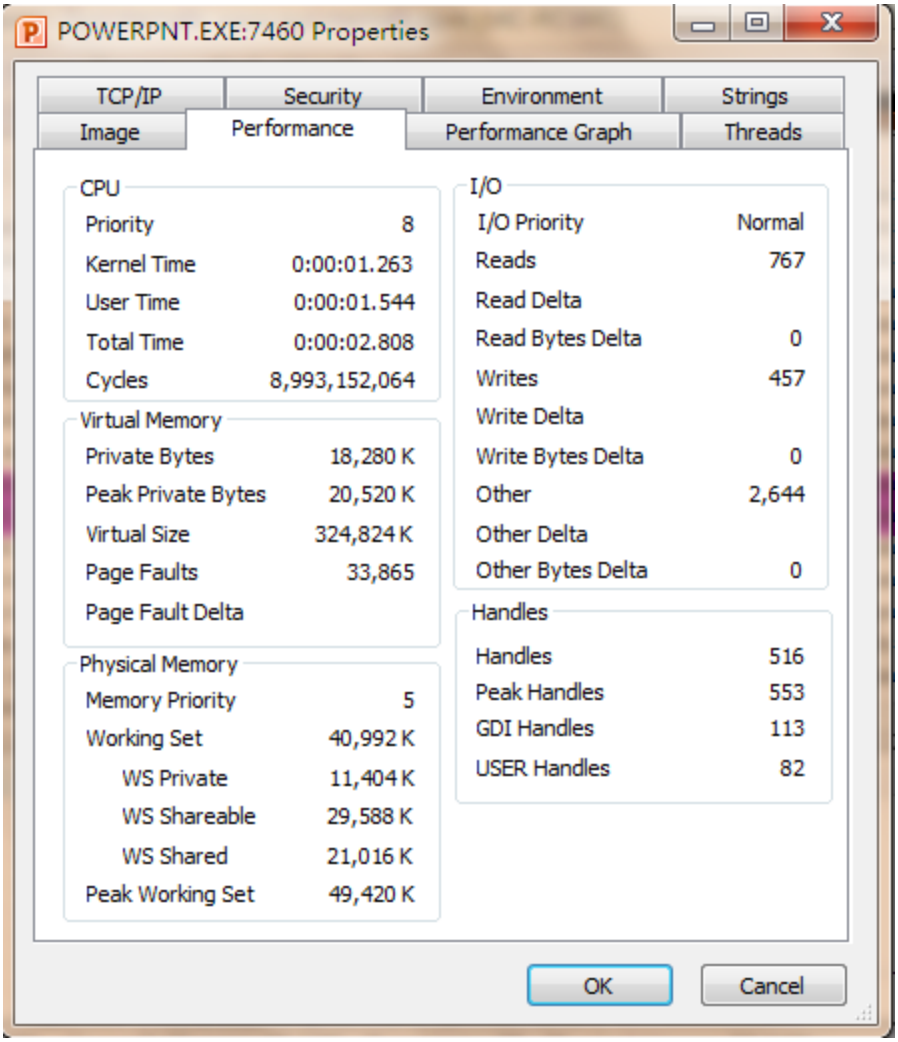
Process Explorer, from *www.sysinternals.com*, shows more details about processes and threads than any other available tool, which is why you will see it used in a number of experiments throughout the book. The following are some of the unique things that Process Explorer shows or enables:

- Full path name for the image being executed
- Process security token (list of groups and privileges)
- Highlighting to show changes in the process and thread list
- List of services inside service-hosting processes, including display name and description
- Processes that are part of a job and job details
- Processes running .NET/WinFX applications and.NET-specific details (such as the list of appdomains and CLR performance counters)
- Start time for processes and threads
- Complete list of memory mapped files (not just DLLs)
- Ability to suspend a process
- Ability to kill an individual thread
- Easy identification of which processes were consuming the most CPU time over a period of time (The Performance Tool can display process CPU utilization for a given set of processes, but it won't automatically show processes created after the performance monitoring session has started.)

We have done some experiments in Process Explorer in the first lab. This time, you are required to do something different.

1. Process Performance

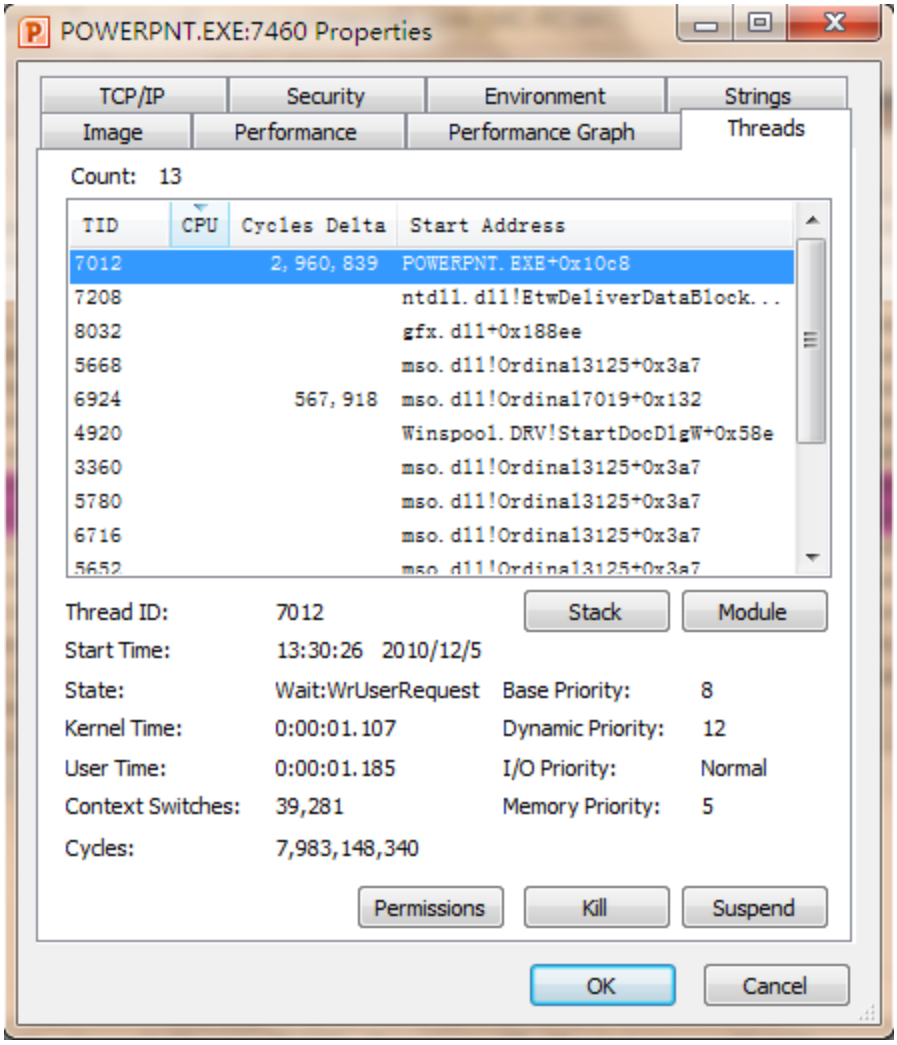
We can use Process Explorer's performance tab to investigate process behavior. Right-click on a process, select **Properties** to open the properties window of that process. Select **Performance** tab. In Process Explorer, you can see more details than Task Manager. Including Handles.



2. Thread Details

In **Thread** tab shows a list of the threads in the process. For each thread it shows the percentage of CPU consumed (based on the refresh interval configured), the number of context switches to the thread, and the thread start address. You can sort by any of these three columns. New threads that are created are highlighted in green, and threads that exit are highlighted in red.

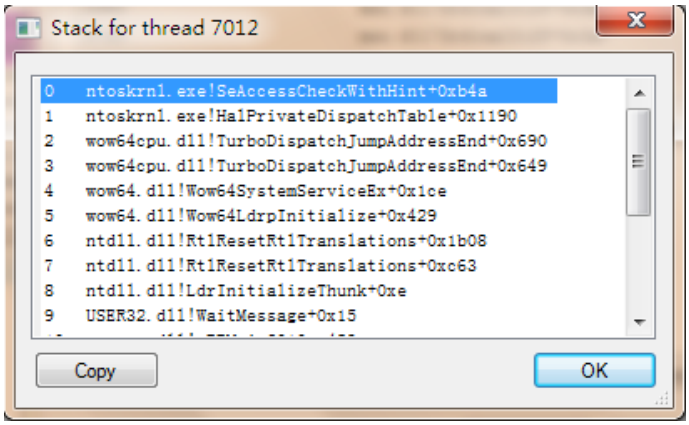
The context switch delta represents the number of times that thread began running in between the refreshes configured for Process Explorer. It provides a different way to determine thread activity than using the percentage of CPU consumed. In some ways it is better because many threads run for such a short amount of time that they are seldom (if ever) the currently running thread when the interval clock timer interrupt occurs, and therefore, are not charged for their CPU time.



3. Thread Start Functions

Notice that, in the above lab, Start Address represents where the thread began running (not where it is now). The thread start address is displayed in the form "*module*!*function*", where *module* is the name of the .exe or .dll. The function name relies on access to symbol files for the module.

If properly configured, Process Explorer can access symbol information to display the symbolic name of the thread start function and functions on its call stack

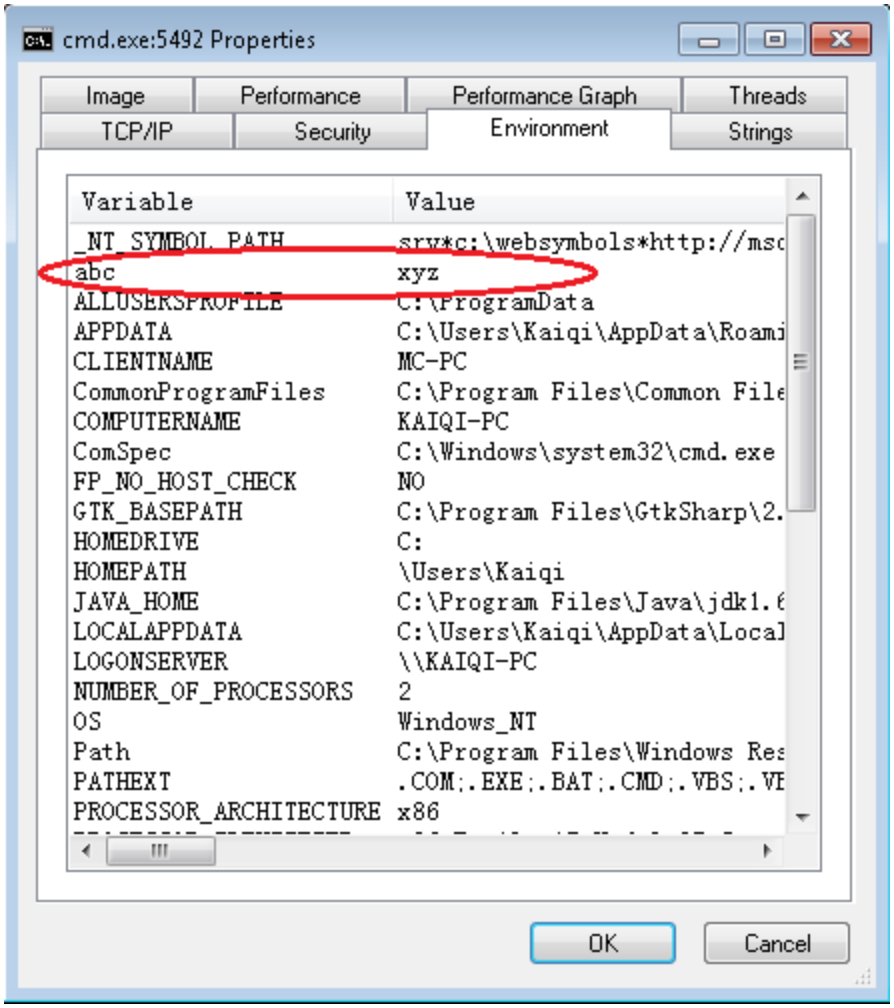


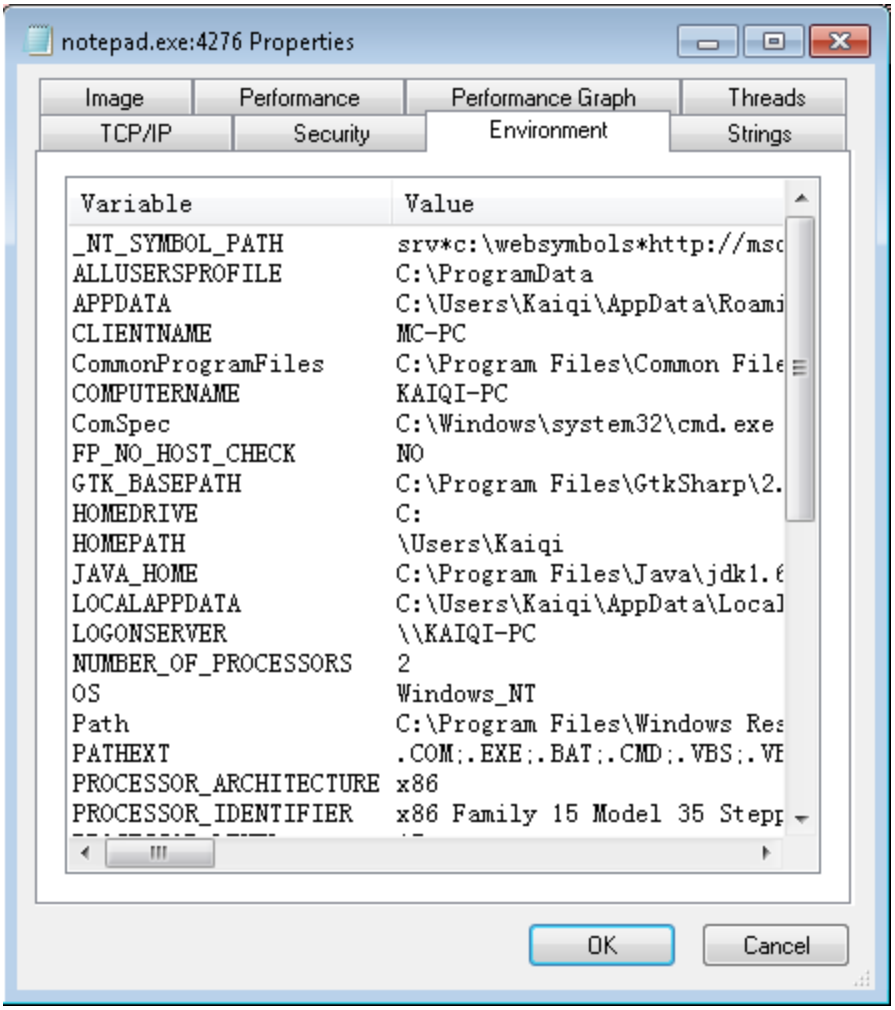
4. Environment Variables

All environment variables of a process can be found in the Environment tab in Process Explorer.

- Open a command prompt
- Run Notepad.exe from command prompt
- Type “set abc=xyz” on the command prompt
- In Process Explorer, examine environment variables for Cmd.exe and Notepad.exe

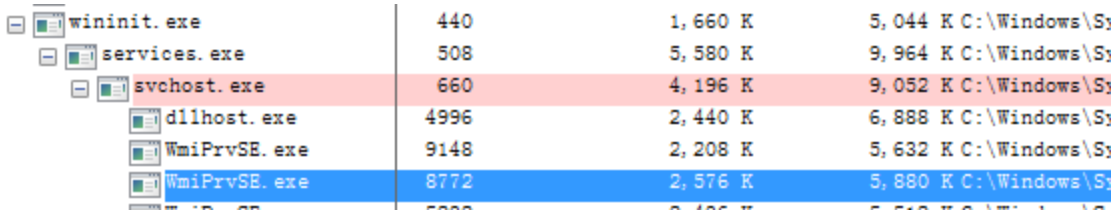
Obviously, The Notepad.exe process didn’t know the environment variable abc, because it is created before setting that variable.

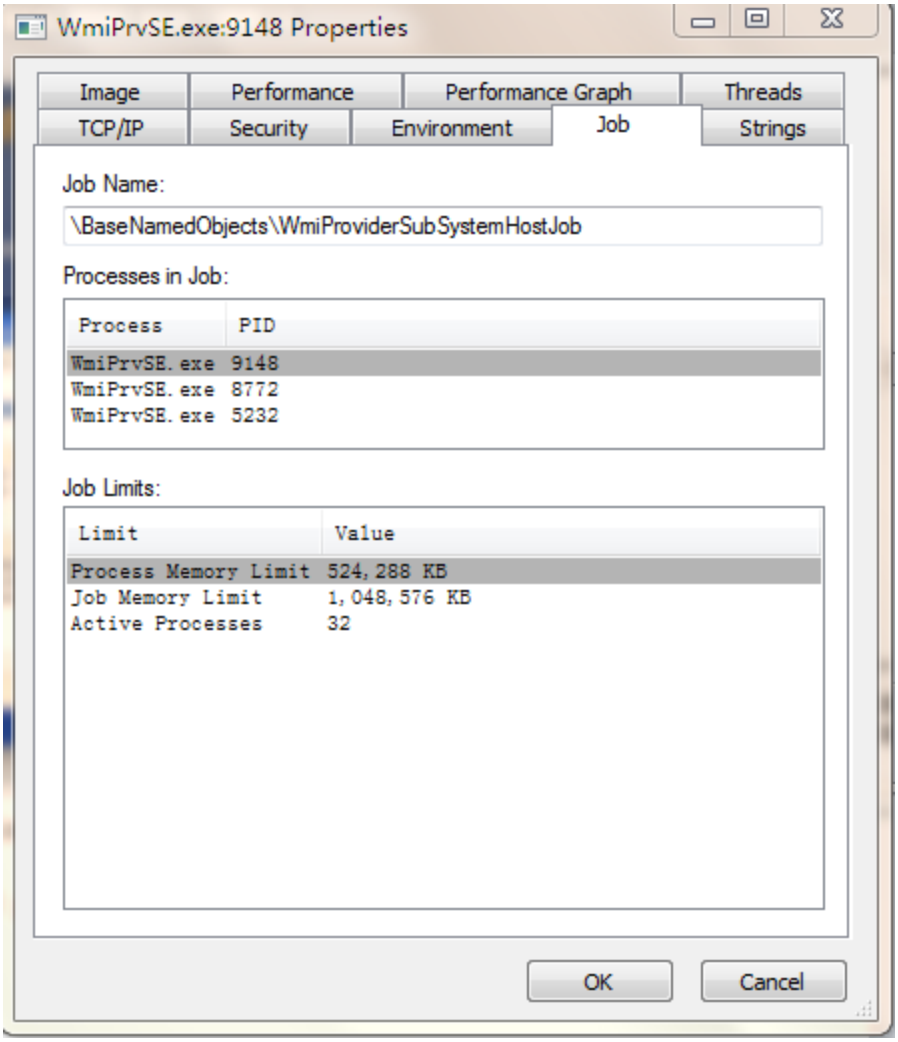




5. Identify Jobs used by WMI

- From a command prompt, run Psinfo (from www.sysinternals.com)
- Notice in Process Explorer two or more WMI (Windows Management Instrumentation) provider processes that are part of a job object (highlighted above)
  (for a description of WMI, see Windows Internals, 4th edition p.237)
- Double click on either Wmiprvse.exe process and click on the **Job** tab.
- Notice the limits set for the job (per-process and job-wide private virtual memory and total active process count)





6. Jobs created by RUNAS

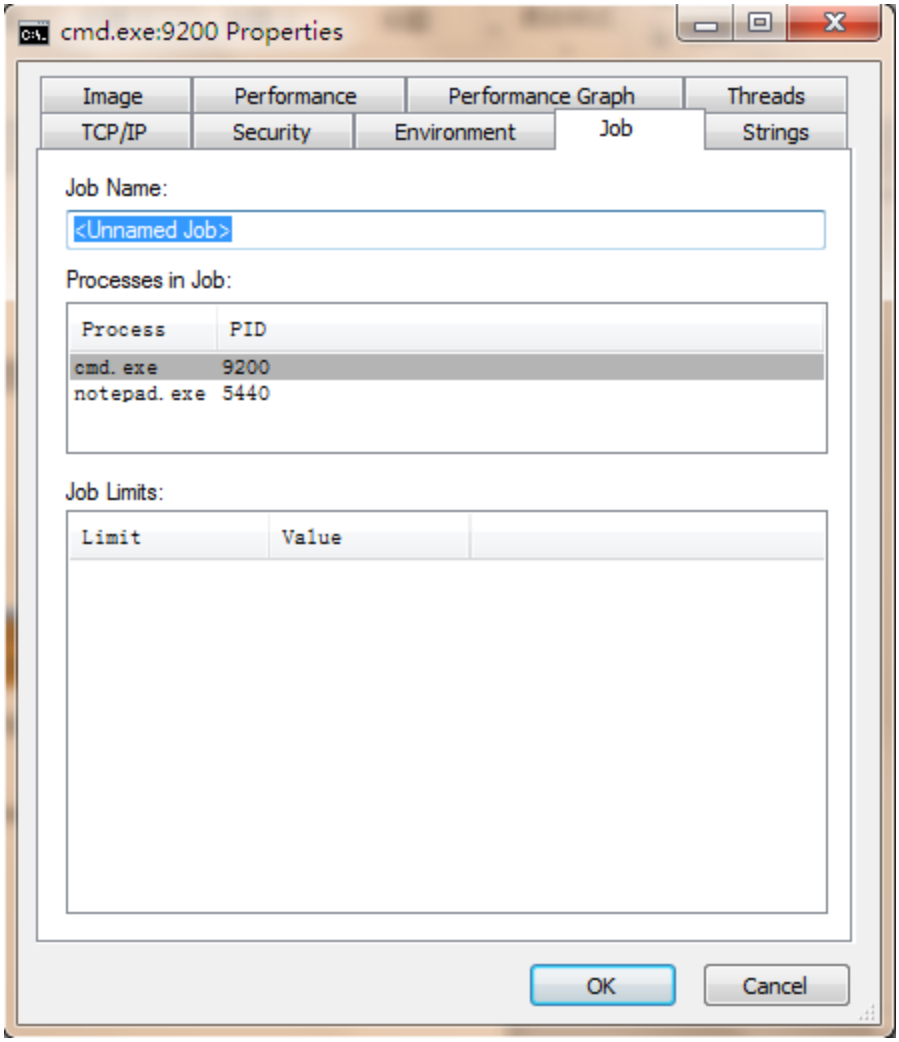
The RUNAS command permits launching processes under alternate credentials. The service behind the RUNAS command (called the Secondary Logon service, and you must enable it before you do this lab) uses a job object to contain the process(es) it creates. This is so that at logoff, the service can terminate all processes that were created by RUNAS and any processes created by these processes, even if the parent/child relationships have been broken.

To view the job object created when RUNAS is used, perform the following steps:

1. From the command prompt, use the *runas* command to create a process running the command prompt (Cmd.exe). For example, type **runas /user:<domain>\< username> cmd**. You'll be prompted for your password. Enter your password, and a command prompt window will appear. The Windows service that executes runas commands creates an unnamed job to contain all processes (so that it can terminate these processes at logoff time).
2. From the command prompt, run Notepad.exe.
3. Then run Process Explorer and notice that the Cmd.exe and Notepad.exe processes are highlighted as part of a job.

4. Double-click either the Cmd.exe or Notepad.exe process to bring up the process properties. You will see a Job tab on the process properties dialog box.

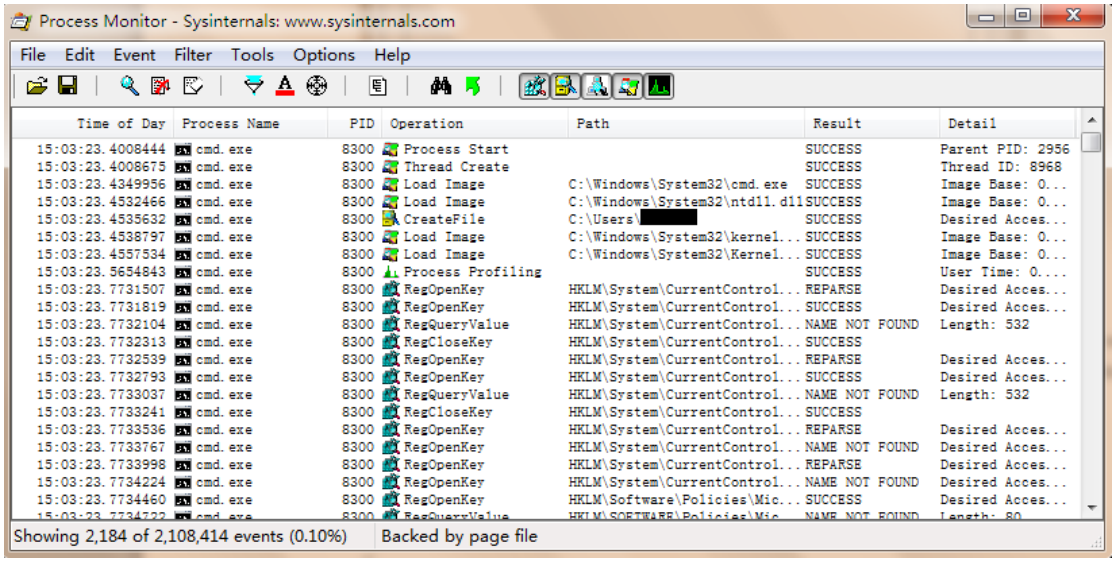
5. Click the **Job** tab to view the details about the job. In this case, there are no quotas associated with the job, but there are two member processes.



Process Monitor

Process Explorer provides many detail information of running process. However, if you want to trace a process, you need to use Process Monitor (*from* www.sysinternals.com). In this lab, you will see what a process do when they run.

- Setup the filter: add two filters with Process Name cmd.exe and notepad.exe.
- Clear the event log and enable **Event Capture** in the **File** Menu.
- Start a command prompt and then open notepad.exe from the prompt.



Process & Thread Blocks

In Kernel Debugger, You can view process(thread) blocks by using !process(!thread) command.

Process Blocks:

The number after “PROCESS” stands for the EPROCESS address. The meaning of other attributes is as follows:

Cid: Process ID.

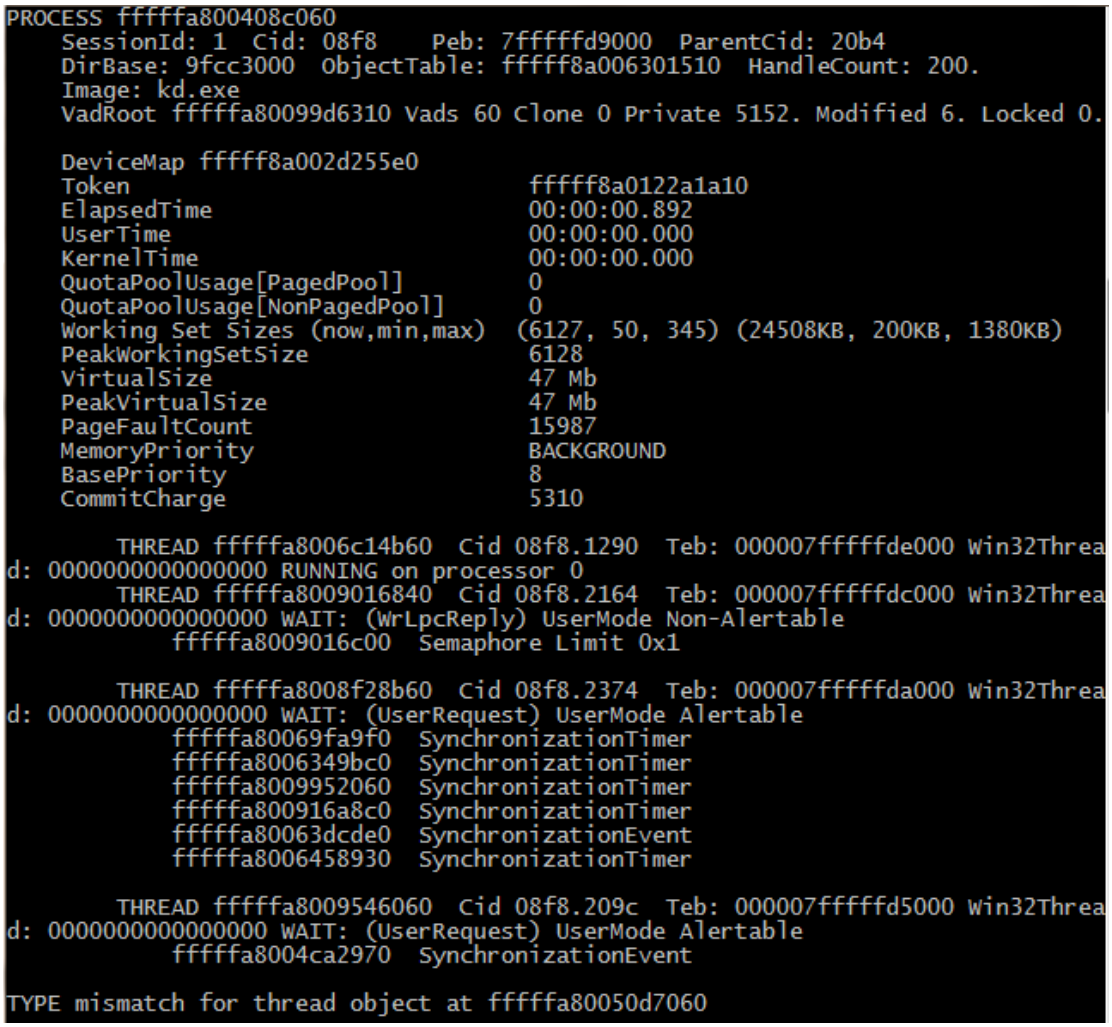
Peb: Address of process environment block.

ParentCid: Process ID of parent process.

DirBase: Physical address of Page Directory.

VadRoot: root of the process’s Virtual Address Descriptor tree.

ElapsedTime: Time the process has been running. It is divided into UserTime and KernelTime.



Thread blocks:

The number after “THREAD” stands for the ETHREAD address. The meaning of other attributes is as follows:

Cid: Process ID . Thread ID.

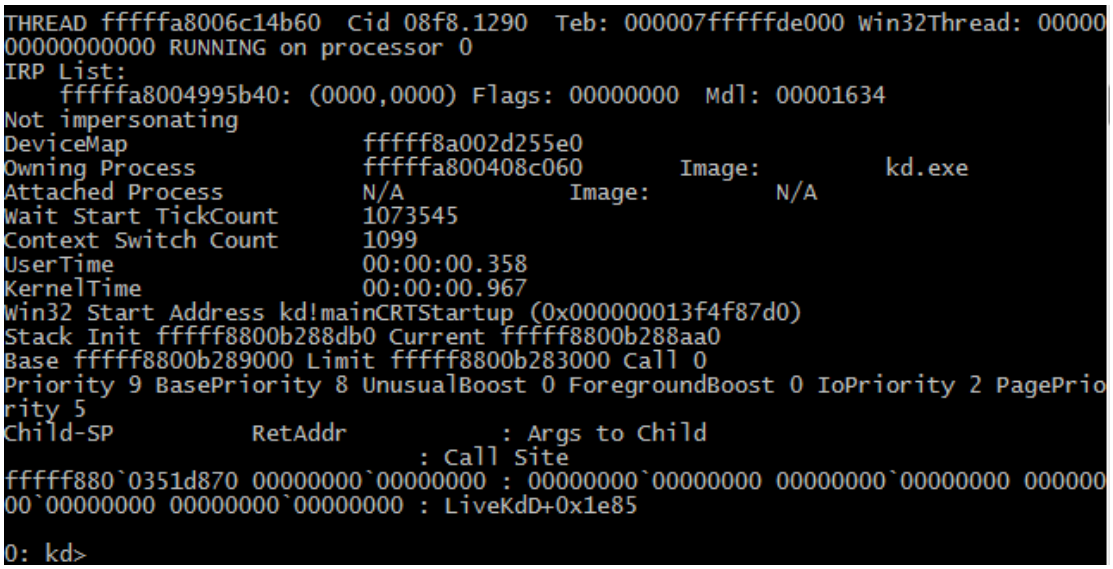
Teb: Address of thread environment block.

Win32Thread: Address of system service dispatch table.

RUNNING/WAIT: Thread state.

Win32 Start Address: Address of user thread function.

Priority, BasePriority: Priority Information.


THREAD fffffa8006c14b60 Cid 08f8.1290 Teb: 000007fffffde000 Win32Thread: 00000
00000000000 RUNNING on processor 0
IRP List:
fffffa8004995b40: (0000,0000) Flags: 00000000 Mdl: 00001634
Not impersonating
DeviceMap fffff8a002d255e0
Owning Process fffffa800408c060 Image: kd.exe
Attached Process N/A Image: N/A
Wait Start TickCount 1073545
Context Switch Count 1099
UserTime 00:00:00.358
KernelTime 00:00:00.967
Win32 Start Address kd!mainCRTStartup (0x000000013f4f87d0)
Stack Init fffff8800b288db0 Current fffff8800b288aa0
Base fffff8800b289000 Limit fffff8800b283000 Call 0
Priority 9 BasePriority 8 UnusualBoost 0 ForegroundBoost 0 IoPriority 2 PagePrio
rity 5
Child-SP RetAddr : Args to Child
: Call Site
fffff880`0351d870 00000000`00000000 : 00000000`00000000 00000000`00000000 000000
00`00000000 00000000`00000000 : LiveKdD+0x1e85
0: kd>