

# LAB 2 PROCESS MANAGEMENT

*PRAMOD NEPAL*

*02/14/16*

*CSCI 5417*

*EAST TENNESSEE STATE UNIVERSITY*

## PURPOSE

The purpose of this lab was to explore some of the process management tools that are available in Windows and Ubuntu.

## MATERIALS

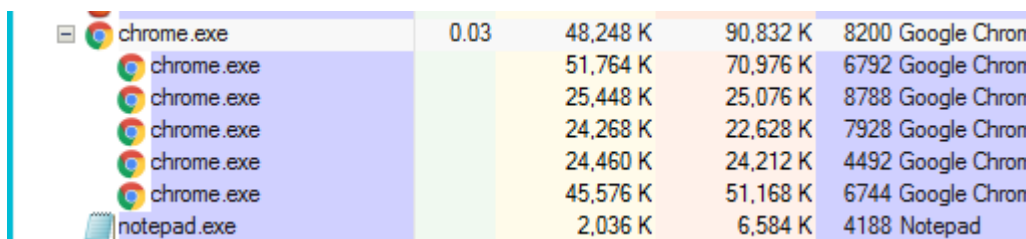
- Lab Instructions
- Web browser
- PuTTY
- AWS console
- SysInternals Process Explorer

## PROCEDURE

### Windows Process Management

To manage processes in Windows we first downloaded the SysInternal suite and started Process Explorer (procexp.exe) as an Administrator. To start an application as an Administrator we can right click the application and select **run as Administrator** menu option. To make this behavior default we can right click the application, select 'Properties', got to the 'Compatibility' tab, check 'Run this program as an administrator' and click Ok.

Once the Process Explorer is running we can show the bottom pane that lists the dynamic link library (dll) associated with a process by pressing **Ctrl-d** key. In Process Explorer, we can see several processes grouped under different names. E.g., In my computer I see several processes under Chrome group. This kind of grouping allows easier management for related tasks. E.g., we can kill all the process started by Chrome by right clicking the group name and selecting **Kill Process Tree** option.



chrome.exe	0.03	48,248 K	90,832 K	8200 Google Chrom
chrome.exe		51,764 K	70,976 K	6792 Google Chrom
chrome.exe		25,448 K	25,076 K	8788 Google Chrom
chrome.exe		24,268 K	22,628 K	7928 Google Chrom
chrome.exe		24,460 K	24,212 K	4492 Google Chrom
chrome.exe		45,576 K	51,168 K	6744 Google Chrom
notepad.exe		2,036 K	6,584 K	4188 Notepad





Next, we open notepad.exe and explore its various properties in the Process Explorer. One thing to note is that the application is grouped under explorer.exe process. Here explorer.exe acts as the User shell, under

which the user initiated processes run. On the bottom pane, if we examine the dll associated with notepad.exe we can see one such file called gdi32.dll. This file provides API for Windows Device Interface. Windows Device Interface enables applications to use graphics and text on video display and printer. We can suspend an application by right clicking the application in the Process Explorer and selecting “Suspend”. Once we suspend an application, it does not respond to mouse interactions. We can make the application functional again by right clicking the application and selecting “Resume” in the Process Explorer. As seen above we can terminate an application by selecting Kill Process from the right click menu. This also removes the application entry from Process Explorer.

In addition to selecting the process from the window, we can also click and drag the circular icon from toolbar that says “Find Window’s Process (drag over window)” to an application to highlight and see information related to that application. E.g., If we drag and drop the icon over Firefox window Process Explorer highlights firefox.exe and shows information related to Firefox.

Next, we launch **cmd.exe** and execute “**ping -t google.com**” to run ping continuously. We then kill cmd.exe from Process Explorer. The cmd window did not close (although the cmd process was killed) and the ping continued to run. To stop the process we can either close the cmd.exe window or press **Ctrl-c** to stop the ping command (Ctrl-c terminates ping and then cmd.exe gets closed). Instead of closing the application from outside window if we explore Process Explorer ping can be found under explorer.exe. This shows that a rouge application attaches itself under the parent of the process that started it. Since cmd was a child of Explorer ping listed itself under Explorer once cmd was killed. A better option would be to use **Kill Process Tree** to kill all the processes under a group.

We also use VirusTotal.com to scan our running process against 50+ separate virus scanners. In my computer I found few applications that showed as as affected. The threats could be a false positive as indicated in the lab instructions, but to on safe side I upgraded some of the applications to newer versions and uninstalled few of them.

 vlc.exe	4.54	45,280 K	39,396 K	8780 VLC media player	VideoLAN	1/54
 Unsigned ThemesSvc.exe		1,824 K	1,036 K	332 Unsigned Themes Service	The Within Network, LLC	0/53
 procexp64.exe	1.30	27,540 K	44,548 K	7400 Sysinternals Process Explorer	Sysinternals - www.sysinternals.com	0/54
 procexp64.exe		2,564 K	7,728 K	6170 Sysinternals Process Explorer	Sysinternals - www.sysinternals.com	0/54

We can also replace Task Manager with Process Explorer so that we can use it to manage tasks in future.

## Linux Process Management

We connect to the Linux OS as instructed in last lab.

### top

Our first task on Linux was to learn about the **top** command. We can launch this program by issuing the **top** command in the terminal window. We then learn about different header names using **man top** command.

```
top - 17:08:11 up 24 min,  0 users,  load average: 0.30, 0.32, 0.36
Tasks: 184 total,  1 running, 183 sleeping,  0 stopped,  0 zombie
%Cpu0  :  6.2/2.7   9[|||||]
%Cpu1  :  7.4/1.3   9[|||||]
%Cpu2  :  4.0/2.7   7[|||||]
%Cpu3  :  2.7/0.7   3[|||||]
GiB Mem : 50.7/3.760
GiB Swap : 0.0/3.813
```

PID	USER	PR	NI	VIRT	RES	%CPU	%MEM	TIME+	S	COMMAND
1	root	20	0	113.8m	4.9m	0.0	0.1	0:00.88	S	systemd
154	root	20	0	30.0m	4.5m	0.0	0.1	0:00.40	S	- systemd-journal
206	root	20	0	35.8m	4.6m	0.0	0.1	0:00.75	S	- systemd-udevd
1527	root	20	0	14.9m	2.4m	0.0	0.1	0:00.31	S	- mount.ntfs
1529	root	20	0	14.8m	1.9m	0.0	0.0	0:00.01	S	- mount.ntfs
1531	root	20	0	14.8m	1.9m	0.0	0.0	0:00.01	S	- mount.ntfs
1534	systemd+	20	0	109.2m	2.9m	0.0	0.1	0:00.01	S	- systemd-timesyn
1538	root	20	0	38.2m	4.3m	0.0	0.1	0:00.02	S	- bluetoothd
1539	mysql	20	0	630.7m	133.2m	0.7	3.5	0:01.88	S	- mysql
1541	root	20	0	15.0m	2.6m	0.0	0.1	0:00.07	S	- systemd-logind
1542	root	20	0	339.1m	14.1m	0.0	0.4	0:01.06	S	- NetworkManager
1695	root	20	0	14.4m	9.4m	0.0	0.2	0:00.01	S	- dhclient
1544	dbus	20	0	35.2m	4.6m	0.0	0.1	0:00.87	S	- dbus-daemon
1566	root	20	0	103.2m	13.3m	0.0	0.3	0:00.04	S	- sddm
1645	root	20	0	352.4m	72.2m	5.3	1.9	0:53.68	S	- Xorg
1792	root	20	0	151.0m	13.4m	0.0	0.3	0:00.04	S	- sddm-helper

### Useful options from above output

**PID:** Task's unique process ID. In Linux whenever a process is created it is given a unique ID by the operating system called the process ID or PID.

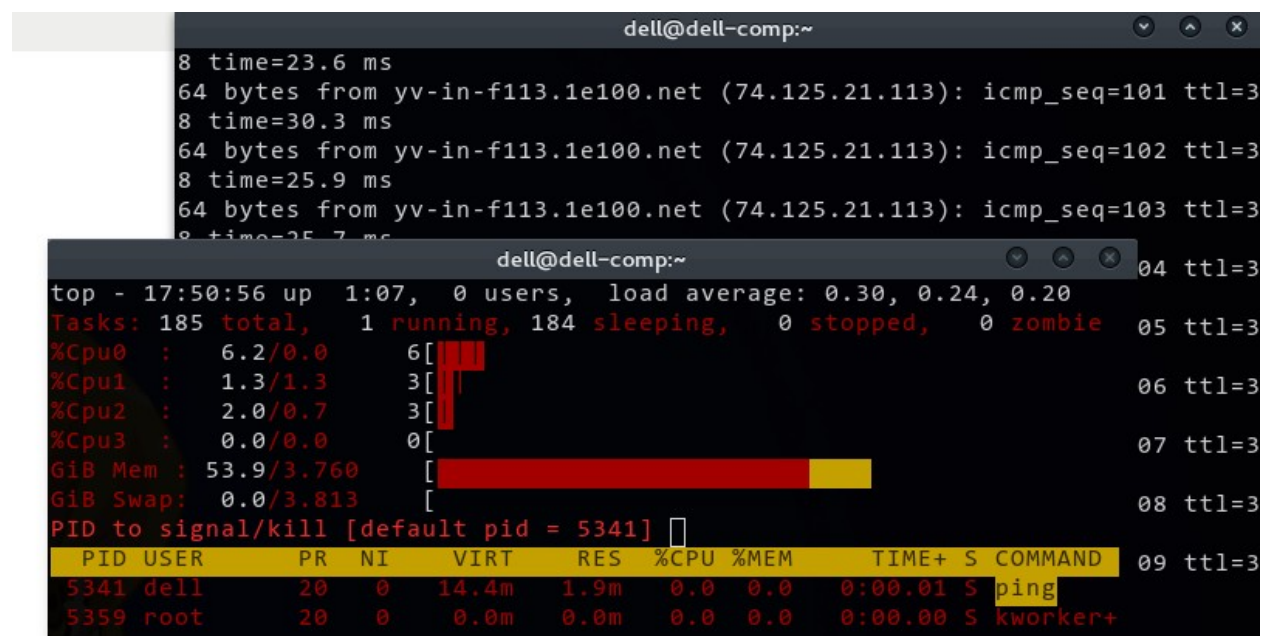
**USER:** Task owner's user name. A process is assigned a user that creates the process. Most of the system services are created as root user. User initiated applications get username as the user.

**PR**: Scheduling priority of the task. Scheduler prioritizes tasks with higher PR values. The job of the scheduler is to share CPU among different processes. Processes with higher priority should get immediate attention, because these tasks could be critical for different system related tasks.

**NI**: The nice value of a task. Nice value is used to prioritize tasks. Negative nice value means higher priority and a positive nice value means lower priority.

We also explore different options and shortcuts in the **top** command. E.g., the 'h' key brings **top**'s help page. This page displays shortcuts and switches for different options. E.g., **L** key can be used to search strings, **z** key can be used to toggle color and monochrome display and **Shift-V** can be used to display forest view (showing process and its dependencies).

We then launch another process by running **ping google.com**. We go back to **top** terminal display, search for the process **ping** using **L** key and type **k** to kill it.



The screenshot shows a terminal window with the following content:

```
dell@dell-comp:~  
8 time=23.6 ms  
64 bytes from yv-in-f113.1e100.net (74.125.21.113): icmp_seq=101 ttl=3  
8 time=30.3 ms  
64 bytes from yv-in-f113.1e100.net (74.125.21.113): icmp_seq=102 ttl=3  
8 time=25.9 ms  
64 bytes from yv-in-f113.1e100.net (74.125.21.113): icmp_seq=103 ttl=3  
8 time=25.7 ms  
04 ttl=3  
top - 17:50:56 up 1:07, 0 users, load average: 0.30, 0.24, 0.20  
Tasks: 185 total, 1 running, 184 sleeping, 0 stopped, 0 zombie  
%Cpu0 : 6.2/0.0 6[  
%Cpu1 : 1.3/1.3 3[  
%Cpu2 : 2.0/0.7 3[  
%Cpu3 : 0.0/0.0 0[  
GiB Mem : 53.9/3.760 [   
GiB Swap: 0.0/3.813 [   
PID to signal/kill [default pid = 5341] [   
PID USER PR NI VIRT RES %CPU %MEM TIME+ S COMMAND 09 ttl=3  
5341 dell 20 0 14.4m 1.9m 0.0 0.0 0:00.01 S ping  
5359 root 20 0 0.0m 0.0m 0.0 0.0 0:00.00 S kworker+
```

This quits the ping command in another terminal.

**ps uax (with a little less and grep)**

Another command we learn on this lab is the **ps** command. We ran **ps aux | less** command. The **ps** command displays a report of a snapshot of current processes. The **a** in the **ps** command lists all the processes, the **u** option lists the UID of the processes and **x** option shows additional processes that is not attached to any terminal. To examine the output one page at a time, we issued **less** command by using a pipe (|) character.

If we example the output we can see that the entry for **top** and **ps aux** shows different TTY entries.

```
dell      7469    2.0  0.0  38956  3928 pts/2    S+   18:51   0:00 top
dell      7470    0.0  0.0  36444  3480 pts/0    R+   18:51   0:00 ps aux
```

The values 0 and 2 indicate the pseudo terminal numbers that the user is logged in. This can also be viewed by executing **who** command.

Next we use **ps aux | grep "www-data"** to list all the process that is under the group www-data. This is helpful for debugging purpose, because it identifies all processes (in this case apache2) under a group. This helps to manage process by using names. E.g., if we stop the service apache2 we can check if any rouge(zombie) processes are still running under that group using above command.

In this lab we also tried to close the running process **top** from another terminal using **sudo kill -15 <pid>** command. We could get the **pid** of **top** command by running **ps aux | grep top** command. A **kill** command terminates the running process. In most situations **top** might be a better option to manage processes. However since there are large number of processes running at one time on Linux, a static dump of currently running processes (**ps aux | grep less**) can be useful when we need to discover processes. We may not always be interested in investigating processes with high resource usage as in **top**.

### **htop (with a little apt-get install)**

Next, we install htop. This is a small utility that is like top but has more features. Htop allows vertical and horizontal scroll, so that we can see all the processes running on the system. It also shows full command line path of the process. Also tasks can be killed, reniced without entering the PIDs. We can install **htop**

using **sudo apt-get install htop -y** command. After installing **htop** we run **ping** command in one terminal and use **htop** to kill the command from another terminal. To search a process in **htop** we could press F3 and use F9 to kill it after it is highlighted.

## **/proc**

Next we investigate the **/proc** directory. This is a pseudo-directory that is created for running process at runtime. We can list the directory using the **ll /proc** command. If we run **ll /proc | less "username"** we can see processes started by that user.

```
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2076/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2080/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2083/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2088/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2114/
dr-xr-xr-x  9 nepalp  nepalp  0 Sep 13 11:30 2130/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2146/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2151/
dr-xr-xr-x  9 nepalp  nepalp  0 Feb 15 18:15 2155/
dr-xr-xr-x  9 nepalp  nepalp  0 Sep 13 11:30 2166/
```

We can also use **cat /proc/cpuinfo** command to see the information about the hardware that Linux is running.

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz
```

Finally, we shutdown the OS by running **sudo shutdown now** command.

## **OBSERVATIONS**

In this Lab we learned about process management in both Windows and Linux Operating Systems. In Windows, we first downloaded the SysInternal suite from Microsoft's website. There are several useful utilities in SysInternal archive, but the one we were interested was Process Explorer. We started the application as an Administrator, so that we have privileges to manage all the processes. We also learned

that in Windows user initiated applications are grouped under Explorer and we can kill all the processes under a group by terminating the group using **Kill Process Tree** option. We also scanned the running processes using an online virus scanner website called VirusTool.com. It scans the process for viruses using 50+ virus scanners. We also found out that some of the results could be false positive.

On Linux side we used several process management commands like **top**, **ps**, and **htop** to manage tasks. Top is a utility to list and manage processes that uses a ncurses based interface. We investigate meanings of several names in **top** window like PID, USER, PR and NI. We can use **man top** to learn the meaning of these names. One notable thing that we learned was that Linux uses PID, which is a unique ID given to a process by the Linux operating system to uniquely identify it. We explore different options used in the **top** command using the help option. We can use L key to search and Shift-V to display forest view. Next command we learned was **ps**. We use **uax** option to enable listing all processes. Since **ps** command lists the snapshot of running process and its output is very large, we use **less** command to list the output one page at a time. One thing we found out while executing **ps** command was that the command lists the terminal number of currently running process. We learned to search process by name e.g., **ps aux | "www-data"** lists processes that are running for apache2 service. We also investigated **kill** command to kill a process using priority value. Finally, we install an application called **htop** using **sudo apt-get install htop**. Htop is an application that has more features than top. We can search a process and kill it using just few keys in htop unlike in top. The commands are listed in htop window.

Our final task was to investigate a pseudo-directory called **/proc**. This directory is created at run-time to list process information and different hardware information. We searched for application started by the logged in user and also found out information about the hardware.

This lab provided us with several useful utilities to investigate processes in Windows and Linux. I believe these commands and tools will be helpful in future labs and in our professional work. Using these tools we can now investigate processes that might be consuming large portion of system resource or behaving badly.