OPERATING SYSTEMS LAB

Submitted by:

Muhammad Umair Kamran (2017-EE-019)
Muhammad Danyal Aman (2017-EE-030)
Tariq Mehmood (2017-EE-040)

Submitted to:

Dr. Nouman Ahmad

L-123 Operating Systems

Spring 2020

Operating Systems

Lab 1

EXERCISE:

QUESTION 1:

(a) Explain the following terms processor and cores.

Processor: A physical chip that contains one or more CPUs. It is defined as the unit which controls and monitors the running activities of the CPU. It is an electronic circuit inside the computer that carries out instruction to perform arithmetic, logical, control and input/output operations

Cores: The basic computation unit of the CPU. It refers to an execution unit inside the CPU that receives and executes instructions.

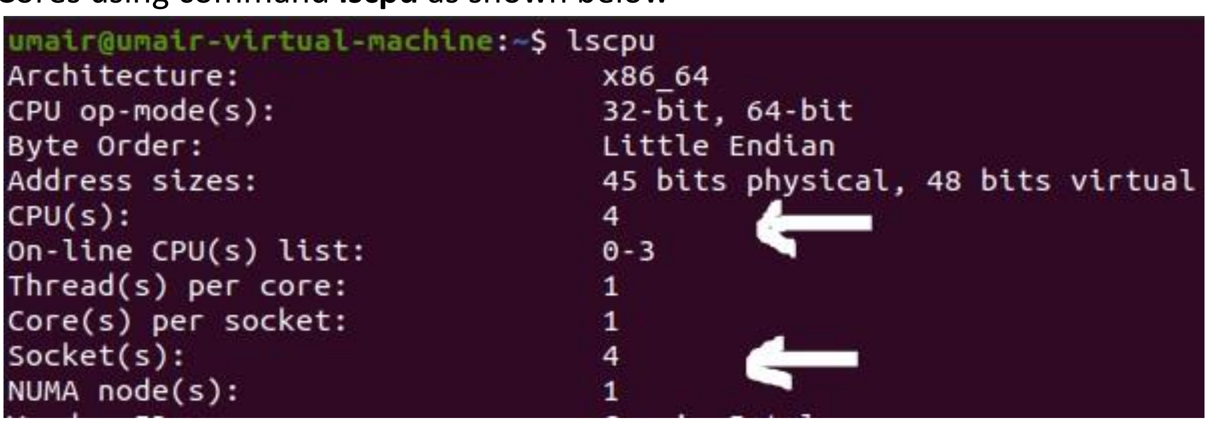
(b) How many cores does your machine have?

There are **4** cores in the machine

c) How many processors does your machine have?

There are 4 processors in my machine.

We used **more /proc/cpuinfo and cat /proc/cpuinfo** to check the information about processors and cores. To verify it we can see the number of Processors and Cores using command **lscpu** as shown below



```
umair@umair-virtual-machine:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   45 bits physical, 48 bits virtual
CPU(s):                          4
On-line CPU(s) list:             0-3
Thread(s) per core:              1
Core(s) per socket:              1
Socket(s):                       4
NUMA node(s):                    1
```

(d) What is the frequency of each processor?

Frequency of each processor can be seen by entering **more /proc/cpuinfo or cat /proc/cpuinfo** commands.
So, the frequency of each processor is **2903.996 MHz** as shown in figure below:

```
processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM)
stepping        : 9
microcode       : 0xb4
cpu MHz         : 2903.996
```

e & f) How much physical memory does your system have much of this memory is free?

We used **free -m** and **cat /proc/meminfo** command to find the physical memory of the system and the amount of it that is free.

```
umair@umair-virtual-machine:~$ free -k
              total        used        free      shared  buff/cache   available
Mem:        8123384     1334756     5833332       53412      955296     6470296
Swap:       2097148           0     2097148
```

```
umair@umair-virtual-machine:~$ cat /proc/meminfo
MemTotal:        8123384 kB
MemFree:         5877664 kB
MemAvailable:    6503400 kB
Buffers:           41696 kB
Cached:           815528 kB
SwapCached:            0 kB
Active:          1212100 kB
Inactive:         509136 kB
```

(g) What is total number of number of forks since the boot in the system?

3416 forks.

```
umair@umair-virtual-machine:~$ vmstat -f
         3416 forks
```

(h) How many context switches has the system performed since bootup?

```
umair@umair-virtual-machine:~$ man proc | grep -n "context switches"
2922:                     The number of context switches that the system underwe
nt.
```

The system underwent **2922** context switches.

Question 2:

- **a) PID is 3732**
- **b) CPU in 99.7% and memory is 0%**
- **c) It is in a running state**

These can be seen in the picture below after executing the file and using top command:

```
top - 20:41:30 up  4:37,  1 user,  load average: 1.05, 0.65, 0.44
Tasks: 302 total,   2 running, 300 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.0 us,  0.2 sy,  0.0 ni, 74.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7933.0 total,   5615.6 free,   1351.5 used,    965.8 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   6263.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3732 umair     20   0    2364    516    452 R  99.7   0.0   3:55.88 cpu
   3749 umair     20   0   20960   4036   3264 R   0.7   0.0   0:00.18 top
    703 root      20   0  249468   8180   6924 S   0.3   0.1  11:03.66 vmtoolsd
   1321 umair     20   0 4420884 263228  99612 S   0.3   3.2  14:03.19 gnome-s+
```

QUESTION 3:

a) ps e- is used… we can also use **pgrep cpu-print** and **pidof cpu-print** commands.

```
4029 pts/0    00:03:40 cpu-print
```

b) Next, find the PIDs of all the ancestors, going back at least 5 generations

```
umair@umair-virtual-machine:~/Downloads/intro-code$  pstree -s -p -a -G -l 4029
systemd,1 splash
  └─systemd,942 --user
      └─gnome-terminal-,4008
          └─bash,4016
              └─cpu-print,4029
```

c) Using this information, can you describe how I/O redirection is being implemented by the shell?

```
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print > /tmp/tmp.txt &
[1] 4363
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print > /tmp/tmp.txt &
[2] 4364
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print > /tmp/tmp.txt &
[3] 4365
```

Mostly all command gives output on screen or take input from keyboard, but in Linux it's possible to send output to file or to read input from file. For e.g., $ ls command gives output to screen; to send output to file of ls give command, $ ls > filename. It means put output of ls command to filename. To output Linuxcommands result to file. In this, cpu-print is giving output to tmp.txt file.

d) Using this information, can you describe how pipes is being implemented by the shell?

```
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print | grep hello &
[1] 1737
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print | grep hello &
[2] 1739
umair@umair-virtual-machine:~/Downloads/intro-code$ ./cpu-print | grep hello &
[3] 1741
```

A pipe is a way to connect the output of one program to the input of another program without any temporary file. A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands (Multiple commands) from same command line. Here the grep command in Linux is a utility used to search any given input files for one or more matching words or patterns "Hello" in this case. It is used to search a single file or an entire directory, including child directories, for a matching string "Hello" in this case.

e) Type of the following commands:

```
umair@umair-virtual-machine:~/Downloads/intro-code$ type ls
ls is aliased to `ls --color=auto'
umair@umair-virtual-machine:~/Downloads/intro-code$ type cd
cd is a shell builtin
umair@umair-virtual-machine:~/Downloads/intro-code$ type history
history is a shell builtin
umair@umair-virtual-machine:~/Downloads/intro-code$ type ps
ps is /usr/bin/ps
```

Internal Commands: cd, history

External Commands: ls, ps

QUESTION 4:

We first loaded our memory1.c file and used **gcc memory1.c -o memory1.** We then executed and ran our C code which gave us the respective PIDs and size of int in the code. We then used ps command to see the virtual space and physical space of our system when running this code. We repeated the same process for memory2.c file and saw that the physical memory changed when memory2.c is executed. The results of our code are as follows:

```
Program : 'memory_1'
---------------------


PID : 2164
Size of int : 4
```

```
umair@umair-virtual-machine:~/Downloads/intro-code$ ps -o vsz 1887
   VSZ
  6284
umair@umair-virtual-machine:~/Downloads/intro-code$ ps -o rss 1887
  RSS
 4852
```

```
Program : 'memory_2'
---------------------


PID : 2179
Size of int : 4
```

```
umair@umair-virtual-machine:~/Downloads/intro-code$ ps -o vsz 2016
   VSZ
  6284
umair@umair-virtual-machine:~/Downloads/intro-code$ ps -o rss 2016
  RSS
 4928
```

OS can allocate virtual memory via page tables / its internal memory book keeping (VSZ virtual memory) before it actually has a backing storage on RAM or disk (RSS resident memory. So, RSS is the total memory taken by a process and indicates that the space taken by memory 2 is more than the code memory1.c

We can also use **sudo pmap** to find out the physical memory which will be same for memory1.c and memory2.c as shown below:

```
umair@umair-virtual-machine:~/Downloads/intro-code$ sudo pmap 2033 | tail -n 1
[sudo] password for umair:
 total             6288K
```

QUESTION 5:

Solution:

Copying foo.pdf 5000 times **using ./make-scripts.sh**. We then ran the process of disk.c and during its run we checked the disk utilization which can be seen in the row of sda as shown below

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.85    0.14    7.37    2.70    0.00   86.94

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_
wrtn    kB_dscd
loop0             0.01         0.07         0.00         0.00        359
   0          0
loop1             0.01         0.20         0.00         0.00       1095
   0          0
loop2             0.01         0.07         0.00         0.00        362
   0          0
loop3             0.01         0.06         0.00         0.00        347
   0          0
loop4             0.01         0.20         0.00         0.00       1073
   0          0
loop5             0.10         2.82         0.00         0.00      15286
   0          0
loop6             0.00         0.01         0.00         0.00         28
   0          0
sda               5.95       117.25       979.91         0.00     634742    530
4805          0
scd0              0.01         0.20         0.00         0.00       1078
   0          0
```

After this we ran the disk1.c process and checked the disk utilization again which is shown below:

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.25    0.02    5.58   13.23    0.00   78.92

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_
wrtn    kB_dscd
loop0             0.00         0.01         0.00         0.00        359
   0          0
loop1             0.00         0.04         0.00         0.00       1095
   0          0
loop2             0.00         0.01         0.00         0.00        362
   0          0
loop3             0.00         0.01         0.00         0.00        347
   0          0
loop4             0.00         0.04         0.00         0.00       1073
   0          0
loop5             0.02         0.73         0.00         0.00      21231
   0          0
loop6             0.00         0.00         0.00         0.00         28
   0          0
sda              17.38      1311.43       217.56         0.00   37894110    628
6369          0
```

So, we can concur that the disk utilized by the file disk.c is less than the disk utilized by the disk1.c process as shown in the kb_read part of sda which shows the 1st SCSI hard drive. Iostat command gives us the details of input output use of disk and its status while running a program or process.

To clear the Disk Buffer Cache memory, we can use **sysctl command** which allows us to delete the page cache, dentaries and inodes. Moreover, we can also use **sync; echo 3 > /proc/sys/vm/drop caches** to delete the Disk Buffer Cache memory of the system. We can also use crontab to schedule the deletion of our Disk Buffer cache memory.