EE-431L: Operating Systems Lab

Lab Report 1: Introduction

Submitted By:

Maaz Afzaal 2017-EE-83

Azeem Azmat 2017-EE-113

Muhammad Aqdas 2017-EE-127

Submitted To: Mr. Nauman Ahmed

Department of Electrical Engineering

University of Engineering and Technology Lahore

Exercise

The answers to all the questions are provided along with their respective screenshots and code.

Q no 1:

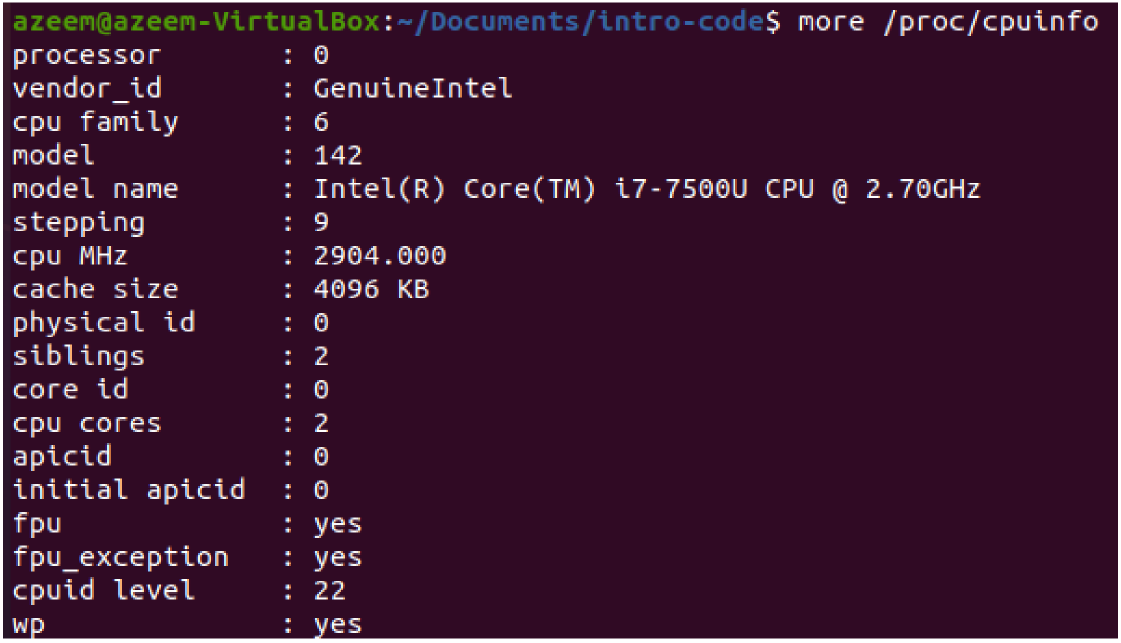
In this question, we will understand the hardware configuration of your working machine using the /proc filesystem.

a) **Run command more /proc/cpuinfo and explain the following terms: processor and cores.**

After running the command

more /proc/cpuinfo

following screen results were observed



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ more /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
stepping        : 9
cpu MHz         : 2904.000
cache size      : 4096 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

A processor is defined as the basic working unit of any machine. It has the ability to take in some input, perform some tasks/processes on the provided input and return the generated output.

An individual processor in a machine is referred to core.

b) **How many cores does your machine have?**

The information provided shows that there are *two* cores.

c) **How many processors does your machine have?**

The information provided shows that there is *no* core.

d) **What is the frequency of each processor?**

The information provided shows that processor frequency is 2904MHz

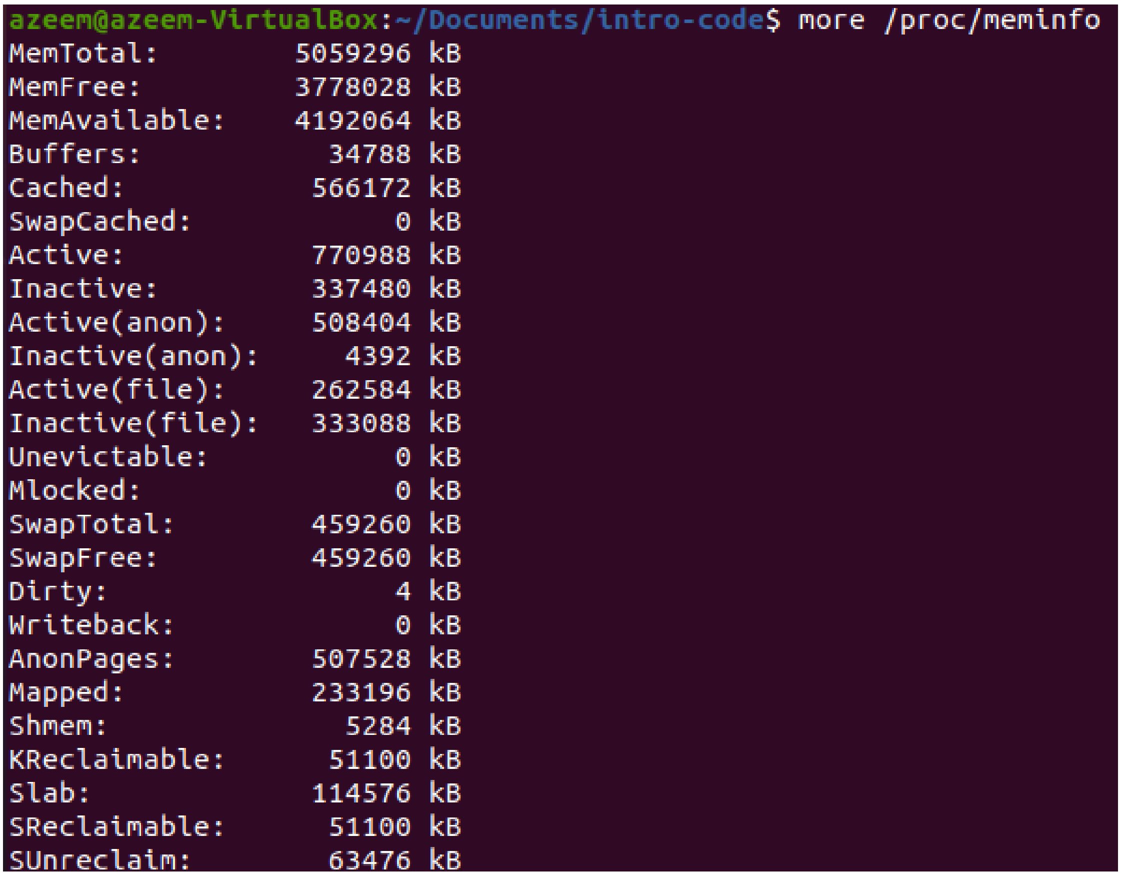
.

e) **How much physical memory does your system have?**

For this part following command was run

more /proc/meminfo

Following information was obtained



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ more /proc/meminfo
MemTotal:        5059296 kB
MemFree:         3778028 kB
MemAvailable:    4192064 kB
Buffers:           34788 kB
Cached:           566172 kB
SwapCached:            0 kB
Active:           770988 kB
Inactive:         337480 kB
Active(anon):     508404 kB
Inactive(anon):     4392 kB
Active(file):     262584 kB
Inactive(file):   333088 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:        459260 kB
SwapFree:         459260 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:        507528 kB
Mapped:           233196 kB
Shmem:              5284 kB
KReclaimable:      51100 kB
Slab:             114576 kB
SReclaimable:      51100 kB
SUnreclaim:        63476 kB
```

According to the information the system has total 5059296 kB of memory.

f) How much of this memory is free?

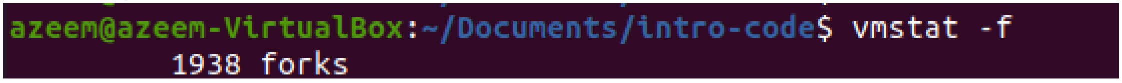
According to the information 3778028 kB of memory is free in the system.

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

For this part following command was run

more /proc/meminfo

Following information was obtained which shows that there are 1938 forks



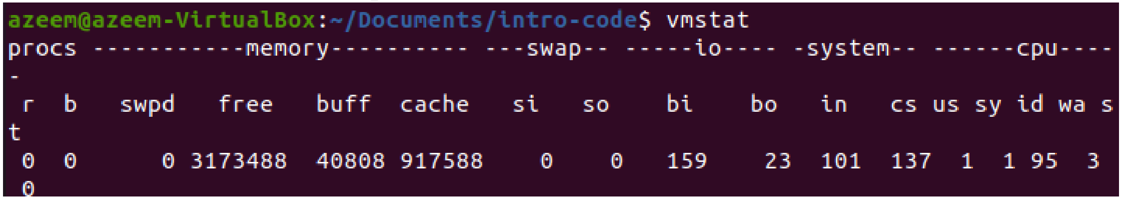
```
azeem@azeem-VirtualBox:~/Documents/intro-code$ vmstat -f
         1938 forks
```

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

For this part a following command was run

vmstat

Following information was obtained which shows the system has performed 137 context switches since bootup.



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu----
-
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa s
t
 0  0      0 3173488  40808 917588    0    0   159    23  101  137  1  1 95  3
 0
```

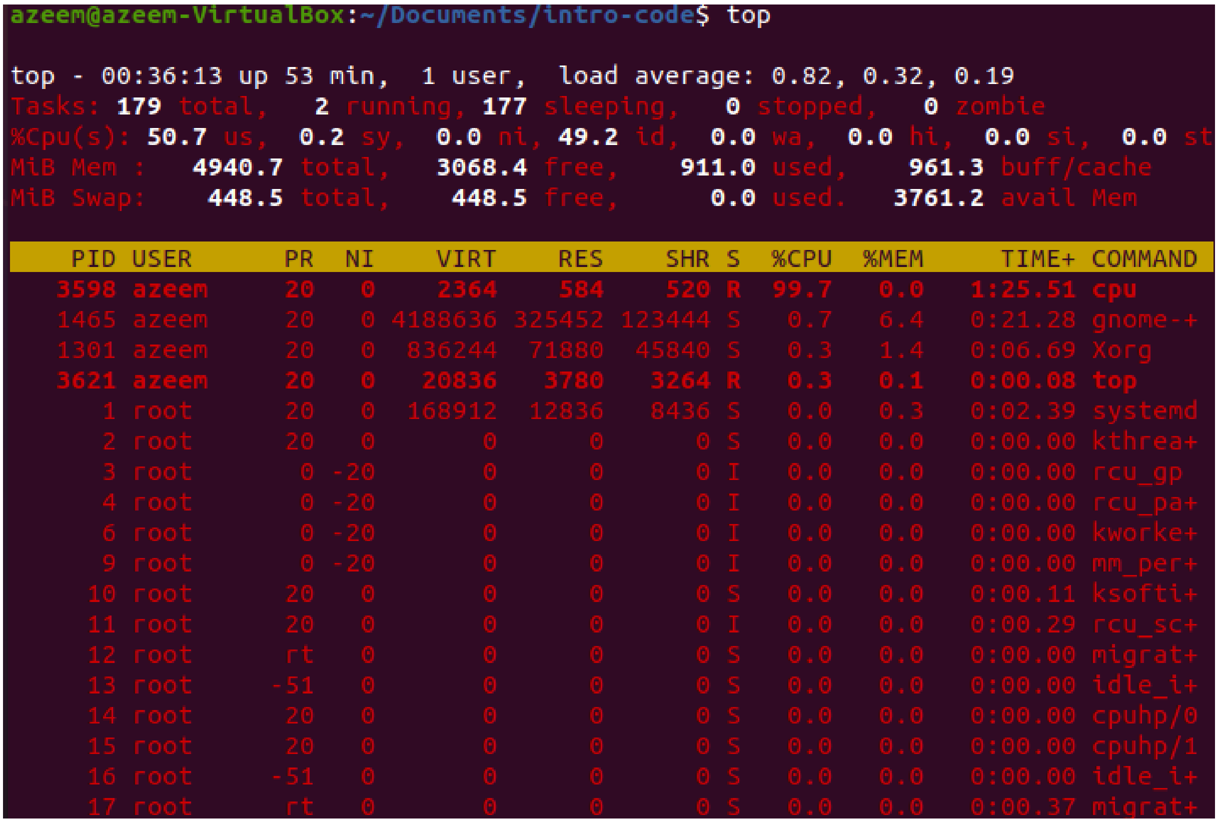
Q no 2:

In this question, we will understand how to monitor the status of a running process using the top command. Compile the program cpu.c given to you and execute it in the bash or any other shell of your choice as follows.

$ gcc cpu.c -o cpu

$ ./cpu

This program runs in an infinite loop without terminating. Now open another terminal, run the top command and answer the following questions about the cpu process.



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ top

top - 00:36:13 up 53 min,  1 user,  load average: 0.82, 0.32, 0.19
Tasks: 179 total,   2 running, 177 sleeping,   0 stopped,   0 zombie
%Cpu(s): 50.7 us,  0.2 sy,  0.0 ni, 49.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   4940.7 total,   3068.4 free,    911.0 used,    961.3 buff/cache
MiB Swap:    448.5 total,    448.5 free,      0.0 used.   3761.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3598 azeem     20   0    2364    584    520 R  99.7   0.0   1:25.51 cpu
   1465 azeem     20   0 4188636 325452 123444 S   0.7   6.4   0:21.28 gnome-+
   1301 azeem     20   0  836244  71880  45840 S   0.3   1.4   0:06.69 Xorg
   3621 azeem     20   0   20836   3780   3264 R   0.3   0.1   0:00.08 top
      1 root      20   0  168912  12836   8436 S   0.0   0.3   0:02.39 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthrea+
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_pa+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworke+
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_per+
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.11 ksofti+
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.29 rcu_sc+
     12 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migrat+
     13 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_i+
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     16 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_i+
     17 root      rt   0       0      0      0 S   0.0   0.0   0:00.37 migrat+
```

a) **What is the PID of the process running the cpu command?**
According to the information the PID of the process running cpu command is 3598

b) **How much CPU and memory does this process consume?**
The process is consuming 99.7% of CPU and 0.0% memory.

c) **What is the current state of the process? For example, is it running or in a blocked state or a zombie state?**
After pressing ‘z’ all the running processes were colored which shows that current state of the process is running.

Q no 3:

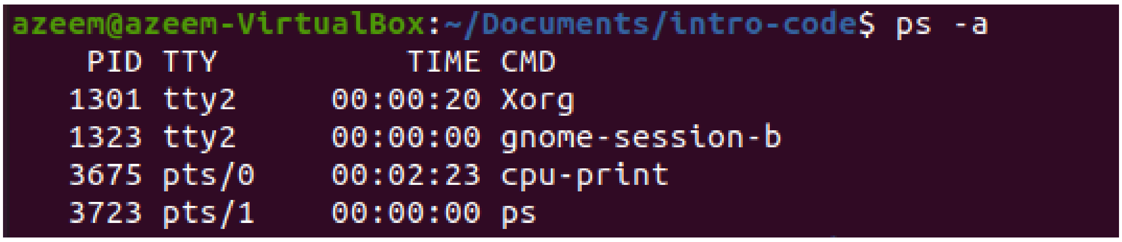
In this question, we will understand how the Linux shell (e.g., the bash shell) runs user commands by spawning new child processes to execute the various commands.

a) **Compile the program cpu-print.c given to you and execute it in the bash or any other shell of your choice as follows.**
$ gcc cpu-print.c -o cpu-print
$ ./cpu-print
This program runs in an infinite loop printing output to the screen. Now, open another terminal and use the ps command with suitable options to find out the pid of the process spawned by the shell to run the cpu-print executable. You may want to explore the ps command thoroughly to understand the various output fields it shows.

Following command was run in order to find the PID of the process.

ps -a

It provided the following results which shows that the PID of the cpu-print process is 3675



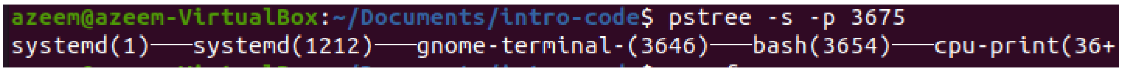
```
azeem@azeem-VirtualBox:~/Documents/intro-code$ ps -a
    PID TTY          TIME CMD
   1301 tty2     00:00:20 Xorg
   1323 tty2     00:00:00 gnome-session-b
   3675 pts/0    00:02:23 cpu-print
   3723 pts/1    00:00:00 ps
```

b) **Find the PID of the parent of the cpu-print process, i.e., the shell process. Next, find the PIDs of all the ancestors, going back at least 5 generations (or until you reach the init process).**

Following command was run to collect the PIDs of all the ancestors of the cpu-print process

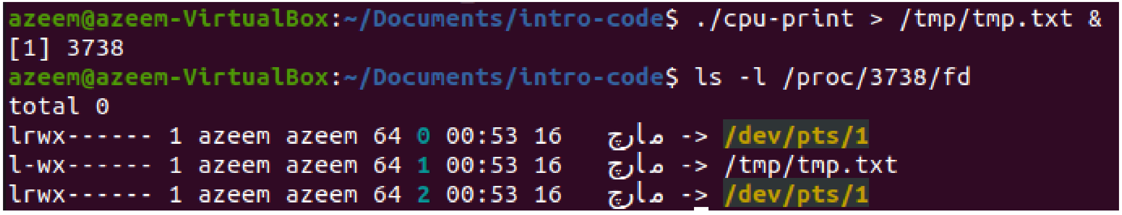
pstree -s -p 3675

The results are shown in the screenshot



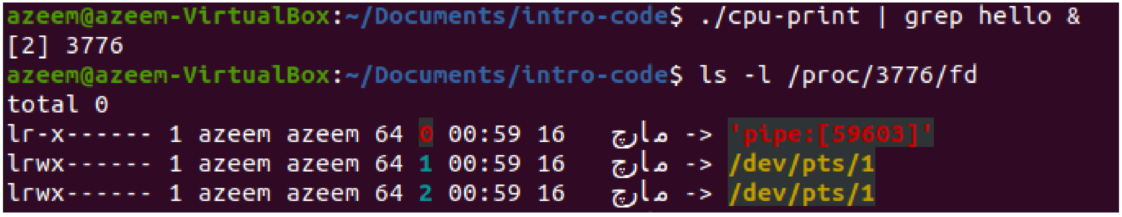
```
azeem@azeem-VirtualBox:~/Documents/intro-code$ pstree -s -p 3675
systemd(1)───systemd(1212)───gnome-terminal-(3646)───bash(3654)───cpu-print(36+
```

c) **We will now understand how the shell performs output redirection. Run the following command.**
./cpu-print > /tmp/tmp.txt &
Look at the proc file system information of the newly spawned process. Pay particular attention to where its file descriptors 0, 1, and 2 (standard input, output, and error) are pointing to. Using this information, can you describe how I/O redirection is being implemented by the shell?



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ ./cpu-print > /tmp/tmp.txt &
[1] 3738
azeem@azeem-VirtualBox:~/Documents/intro-code$ ls -l /proc/3738/fd
total 0
lrwx------ 1 azeem azeem 64 0 00:53 16   ماړچ -> /dev/pts/1
l-wx------ 1 azeem azeem 64 1 00:53 16   ماړچ -> /tmp/tmp.txt
lrwx------ 1 azeem azeem 64 2 00:53 16   ماړچ -> /dev/pts/1
```

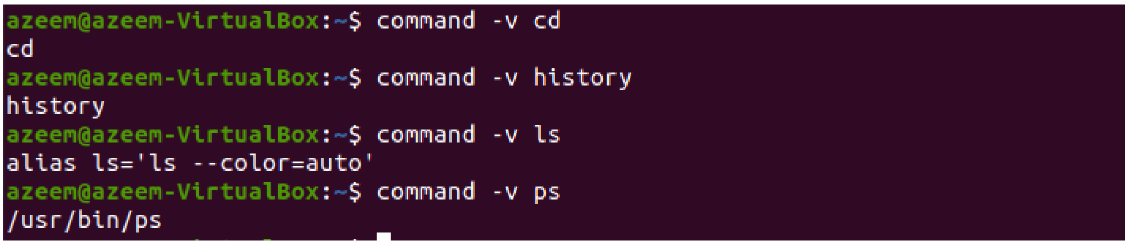
d) **Next, we will understand how the shell implements pipes. Run the following command.**
./cpu-print | grep hello &
Once again, identify the newly spawned processes, and find out where their standard input/output/error file descriptors are pointing to. Use this information to explain how pipes are implemented by the shell.



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ ./cpu-print | grep hello &
[2] 3776
azeem@azeem-VirtualBox:~/Documents/intro-code$ ls -l /proc/3776/fd
total 0
lr-x------ 1 azeem azeem 64 0 00:59 16   مارچ -> 'pipe:[59603]'
lrwx------ 1 azeem azeem 64 1 00:59 16   مارچ -> /dev/pts/1
lrwx------ 1 azeem azeem 64 2 00:59 16   مارچ -> /dev/pts/1
```

e) **When you type in a command into the shell, the shell does one of two things. For some commands, executables that perform that functionality already come built into the Linux kernel. For such commands, the shell simply invokes the executable like it runs the executables of your own programs. For other commands where the executable does not exist, the shell implements the command itself within its code. Consider the following commands that you can type in the bash shell: cd, ls, history, ps. Which of these commands already exist as built-in executables in the Linux kernel that are then simply executed by the bash shell, and which are implemented by the bash code itself?**

Following code was run to determine which is the built-in executable command in Linux kernel and which are implemented by bash code itself
command -v “commandname”



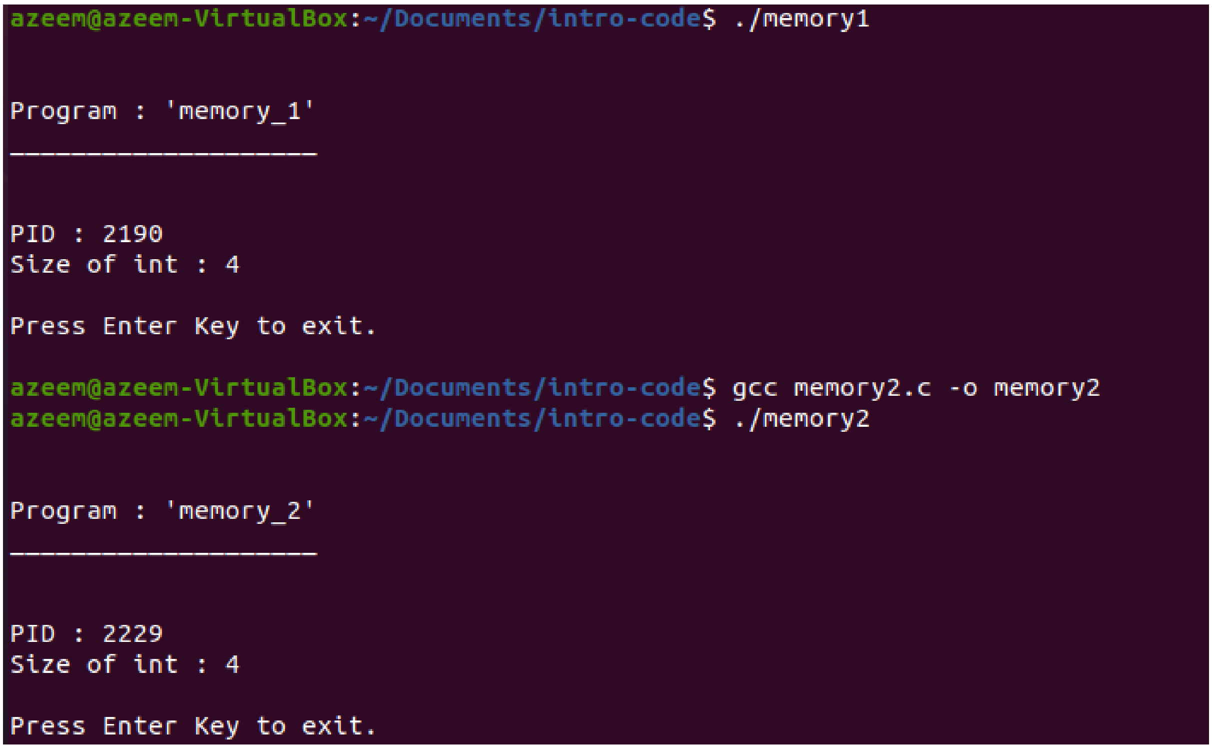
```
azeem@azeem-VirtualBox:~$ command -v cd
cd
azeem@azeem-VirtualBox:~$ command -v history
history
azeem@azeem-VirtualBox:~$ command -v ls
alias ls='ls --color=auto'
azeem@azeem-VirtualBox:~$ command -v ps
/usr/bin/ps
```

The observations clear that **cd** and **history** are the built-in commands while **ls** and **ps** are implemented using the bash code.

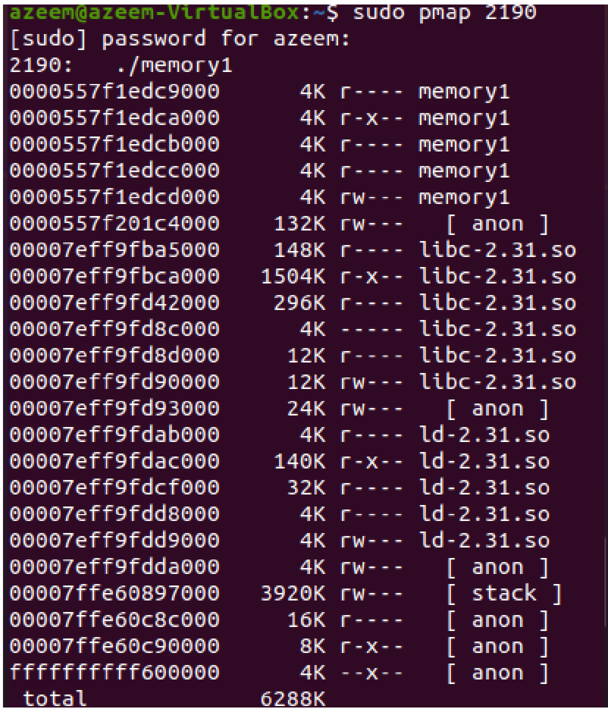
Q no 4:

Consider the two programs memory1.c and memory2.c given to you. Compile and run them one after the other. Both programs allocate a large array in memory. One of them accesses the array and the other doesn’t. Both programs pause before exiting to let you inspect their memory usage. You can inspect the memory used by a process with the ps command. In particular, the output will tell you what the total size of the “virtual” memory of the process is, and how much of this is actually physically resident in memory. You will learn later that the virtual memory of the process is the memory the process thinks it has, while the OS only allocates a subset of this memory physically in RAM. Compare the virtual and physical memory usage of both programs.

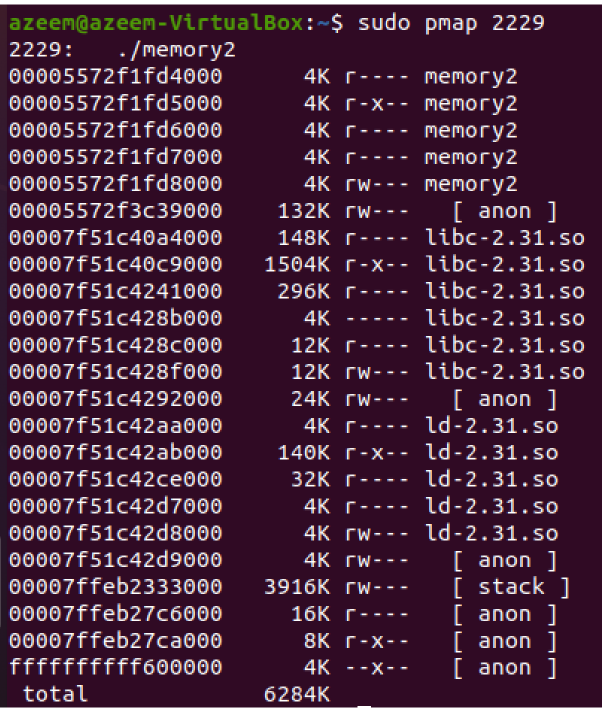
Following commands were run in order compile and run both the programs



```
azeem@azeem-VirtualBox:~/Documents/intro-code$ ./memory1


Program : 'memory_1'
____________________


PID : 2190
Size of int : 4

Press Enter Key to exit.

azeem@azeem-VirtualBox:~/Documents/intro-code$ gcc memory2.c -o memory2
azeem@azeem-VirtualBox:~/Documents/intro-code$ ./memory2


Program : 'memory_2'
____________________


PID : 2229
Size of int : 4

Press Enter Key to exit.
```



```
azeem@azeem-VirtualBox:~$ sudo pmap 2190
[sudo] password for azeem:
2190:   ./memory1
0000557f1edc9000      4K r---- memory1
0000557f1edca000      4K r-x-- memory1
0000557f1edcb000      4K r---- memory1
0000557f1edcc000      4K r---- memory1
0000557f1edcd000      4K rw--- memory1
0000557f201c4000    132K rw---   [ anon ]
00007eff9fba5000    148K r---- libc-2.31.so
00007eff9fbca000   1504K r-x-- libc-2.31.so
00007eff9fd42000    296K r---- libc-2.31.so
00007eff9fd8c000      4K ----- libc-2.31.so
00007eff9fd8d000     12K r---- libc-2.31.so
00007eff9fd90000     12K rw--- libc-2.31.so
00007eff9fd93000     24K rw---   [ anon ]
00007eff9fdab000      4K r---- ld-2.31.so
00007eff9fdac000    140K r-x-- ld-2.31.so
00007eff9fdcf000     32K r---- ld-2.31.so
00007eff9fdd8000      4K r---- ld-2.31.so
00007eff9fdd9000      4K rw--- ld-2.31.so
00007eff9fdda000      4K rw---   [ anon ]
00007ffe60897000   3920K rw---   [ stack ]
00007ffe60c8c000     16K r----   [ anon ]
00007ffe60c90000      8K r-x--   [ anon ]
ffffffffff600000      4K --x--   [ anon ]
 total             6288K
```



```
azeem@azeem-VirtualBox:~$ sudo pmap 2229
2229:   ./memory2
00005572f1fd4000      4K r---- memory2
00005572f1fd5000      4K r-x-- memory2
00005572f1fd6000      4K r---- memory2
00005572f1fd7000      4K r---- memory2
00005572f1fd8000      4K rw--- memory2
00005572f3c39000    132K rw---   [ anon ]
00007f51c40a4000    148K r---- libc-2.31.so
00007f51c40c9000   1504K r-x-- libc-2.31.so
00007f51c4241000    296K r---- libc-2.31.so
00007f51c428b000      4K ----- libc-2.31.so
00007f51c428c000     12K r---- libc-2.31.so
00007f51c428f000     12K rw--- libc-2.31.so
00007f51c4292000     24K rw---   [ anon ]
00007f51c42aa000      4K r---- ld-2.31.so
00007f51c42ab000    140K r-x-- ld-2.31.so
00007f51c42ce000     32K r---- ld-2.31.so
00007f51c42d7000      4K r---- ld-2.31.so
00007f51c42d8000      4K rw--- ld-2.31.so
00007f51c42d9000      4K rw---   [ anon ]
00007ffeb2333000   3916K rw---   [ stack ]
00007ffeb27c6000     16K r----   [ anon ]
00007ffeb27ca000      8K r-x--   [ anon ]
ffffffffff600000      4K --x--   [ anon ]
 total             6284K
```