Lab Report #1

Ahmer Ayaz	**2017-EE-79**
Abdullah Ayaz	**2017-EE-178**
Fahad Ishfaq	**2017-EE-192**

Section D

Nouman Ahmed

Operating Systems

15-03-2021

Department of Electrical Engineering

University of Engineering & Technology, Lahore

Question 1:

```
fahad@fahad-VirtualBox:~/Desktop$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          2
On-line CPU(s) list:             0,1
Thread(s) per core:              1
Core(s) per socket:              2
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           142
Model name:                      Intel(R) Core(TM) m3-7Y30 CPU @ 1.00GHz
Stepping:                        9
CPU MHz:                         1607.999
BogoMIPS:                        3215.99
Hypervisor vendor:               KVM
Virtualization type:             full
L1d cache:                       64 KiB
L1i cache:                       64 KiB
L2 cache:                        512 KiB
L3 cache:                        8 MiB
NUMA node0 CPU(s):               0,1
```

(b) My machine has 2 cores

(c) My machine has two processors, 0 and 1

(d) Each processor has frequency of 1607.999 MHz

```
fahad@fahad-VirtualBox:~/Desktop$ more /proc/meminfo
MemTotal:        4030176 kB
MemFree:         1714836 kB
MemAvailable:    3068680 kB
```

(e)My system has total memory of 4030176 KB (3.843 GB)

(f) 1714836 KB or (1.635 GB) free memory

```
fahad@fahad-VirtualBox:/proc$ more /proc/stat
cpu  33856 12830 9898 1059195 18755 0 842 0 0 0
cpu0 16487 7091 5173 534737 9004 0 396 0 0 0
cpu1 17368 5739 4725 524457 9751 0 445 0 0 0
intr 3679064 29 1453 0 0 0 0 0 0 0 0 0 0 654 0 0 5845 0 0 9509 14237 13001 5050
4 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ctxt 2120529
btime 1615821972
processes 19423
procs_running 2
procs_blocked 0
softirq 3319447 3 1559196 1146 14472 51759 0 1203 1451489 0 240179
```

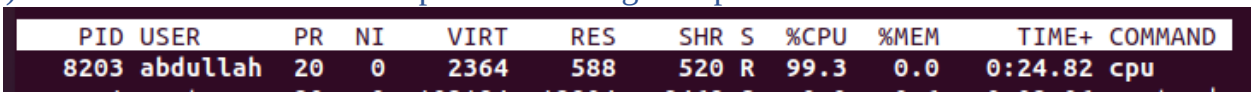
Question 2:

top command is used to show the Linux processes. It provides a dynamic real-time view of the running system.

The information which is displayed using this command is

1. PID: Shows task's unique process id.
2. PR: Stands for priority of the task.
3. SHR: Represents the amount of shared memory used by a task.
4. VIRT: Total virtual memory used by the task.
5. USER: User name of owner of task.
6. %CPU: Represents the CPU usage.
7. TIME+: CPU Time, the same as 'TIME', but reflecting more granularity through hundredths of a second.
8. SHR: Represents the Shared Memory size (kb) used by a task.
9. NI: Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
10. %MEM: Shows the Memory usage of task.

(a) What is the PID of the process running the cpu command?



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8203	abdullah	20	0	2364	588	520	R	99.3	0.0	0:24.82	cpu

Here it can be seen that the PID of cpu command is 8203.

(b) How much CPU and memory does this process consume?

This process consumes 99.3% of the cpu and the amount of shared memory used is 0.

(c) (c) What is the current state of the process? For example, is it running or in a blocked state or a zombie state?

The process is running as the code is simple while loop which never stops running, therefore it is a running process.

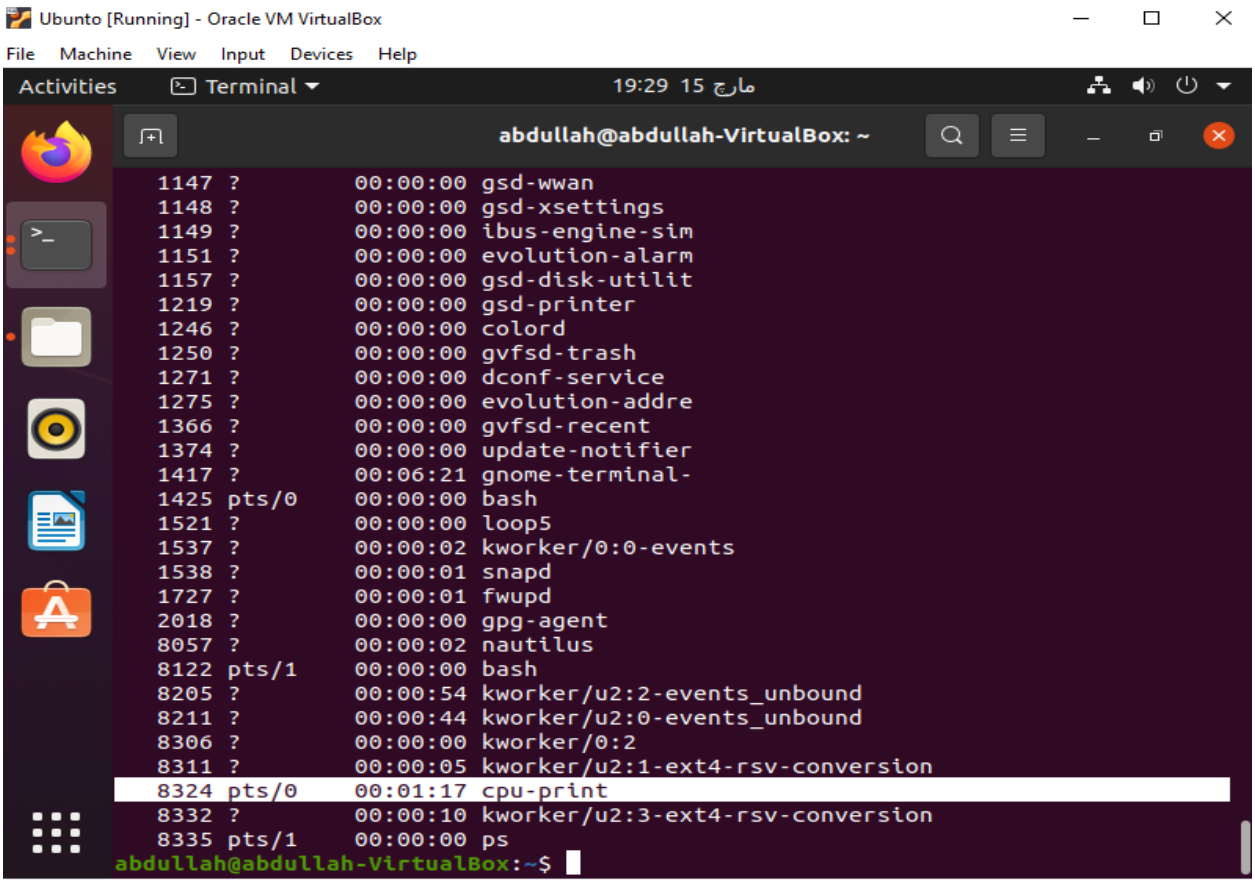
QUESTION 3

(d) find out the pid of the process spawned by the shell to run the cpu-print executable:

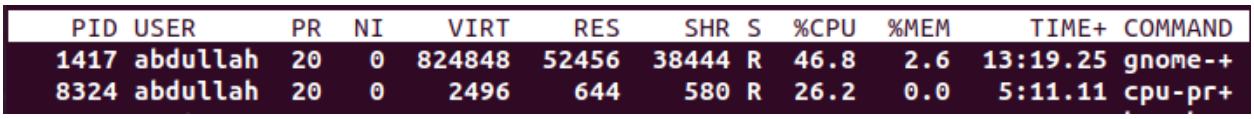
command

ps-A

output



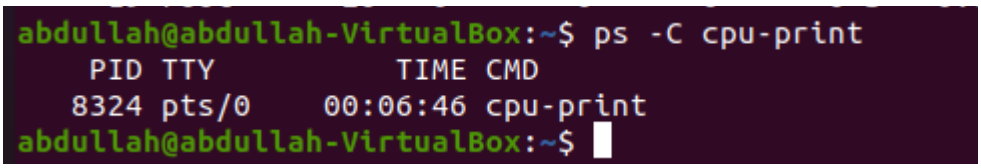
This PID can be crosschecked with ” top “ command, which shows the number 8324 as well.



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1417	abdullah	20	0	824848	52456	38444	R	46.8	2.6	13:19.25	gnome-+
8324	abdullah	20	0	2496	644	580	R	26.2	0.0	5:11.11	cpu-pr+

Another way is using the command:

Ps -C cpu-print



(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).

Command: pstree -s pid

Pid is the process id of the child process we are interested in, it is 8324 in out case.

```
abdullah@abdullah-VirtualBox:~$ pstree -s 8324
systemd---systemd---gnome-terminal-+---bash---cpu-print
abdullah@abdullah-VirtualBox:~$
```

(c) We will now understand how the shell performs output redirection.

standard redirection:

Linux includes redirection commands for each stream. These commands write standard output to a file. If a non-existent file is targetted (either by a single-bracket or double-bracket command), a new file with that name will be created prior to writing.

Commands with a single bracket *overwrite* the destination's existing contents.

```
abdullah@abdullah-VirtualBox:~/Documents/intro-code$ ./cpu-print>/tmp/tmp.txt &
[1] 8748
```

Command for seeing the redirection:

```
ls -l /proc/`pidof cpu-print`/fd
```

```
abdullah@abdullah-VirtualBox:~$ ls -l /proc/`pidof cpu-print`/fd
total 0
lrwx------ 1 abdullah abdullah 64 0 22:26 15   ماج -> /dev/pts/0
lrwx------ 1 abdullah abdullah 64 1 22:26 15   ماج -> /dev/pts/0
lrwx------ 1 abdullah abdullah 64 2 22:26 15   ماج -> /dev/pts/0
```

0 means input, 1 mean output and 2 means error

(d)Next, we will understand how the shell implements pipes.

Pipes are used to redirect a stream from one program to another. When a program's standard output is sent to another through a pipe, the first program's data, which is received by the second program, will not be displayed on the terminal. Only the filtered data returned by the second program will be displayed.

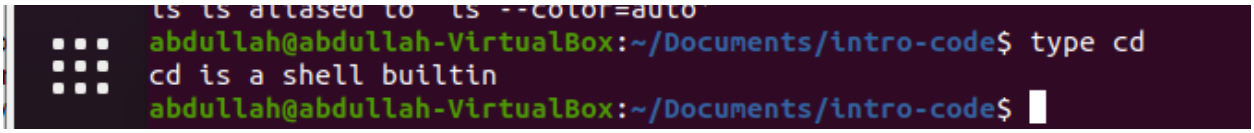
The Linux *pipe* is represented by a vertical bar.

Though the functionality of the pipe may appear to be similar to that of **>** and **>>** (standard output redirect), the distinction is that pipes redirect data from one command to another, while **>** and **>>** are used to redirect exclusively to files

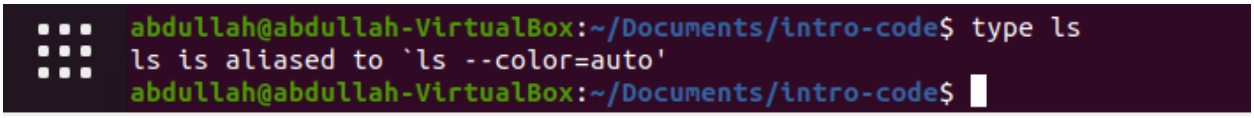
In this the word hello is found from the cpu-print and displayed on the screen.

(e)types of commands:

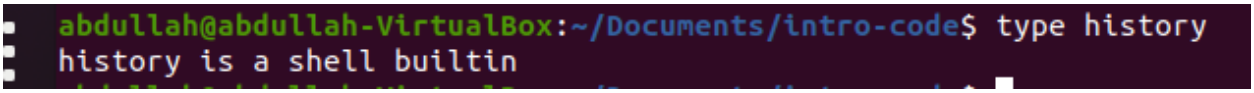
Cd is built in.



Ls is implemented by bash code itself.



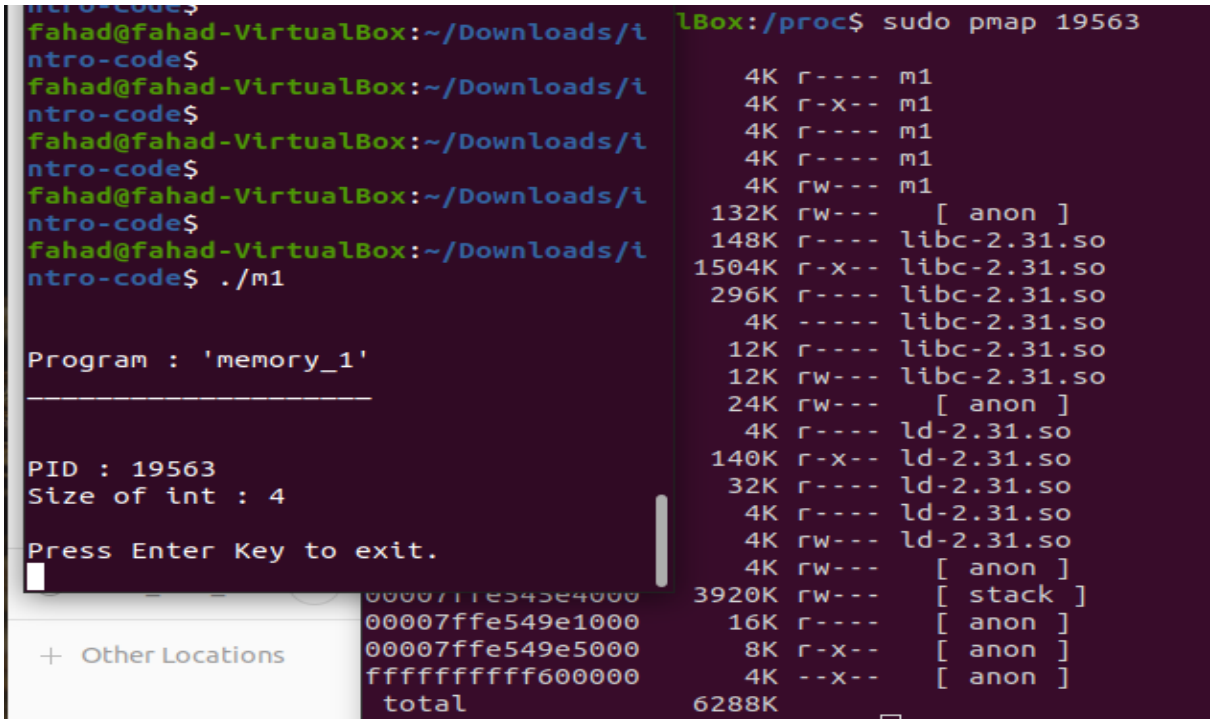
History is built-in



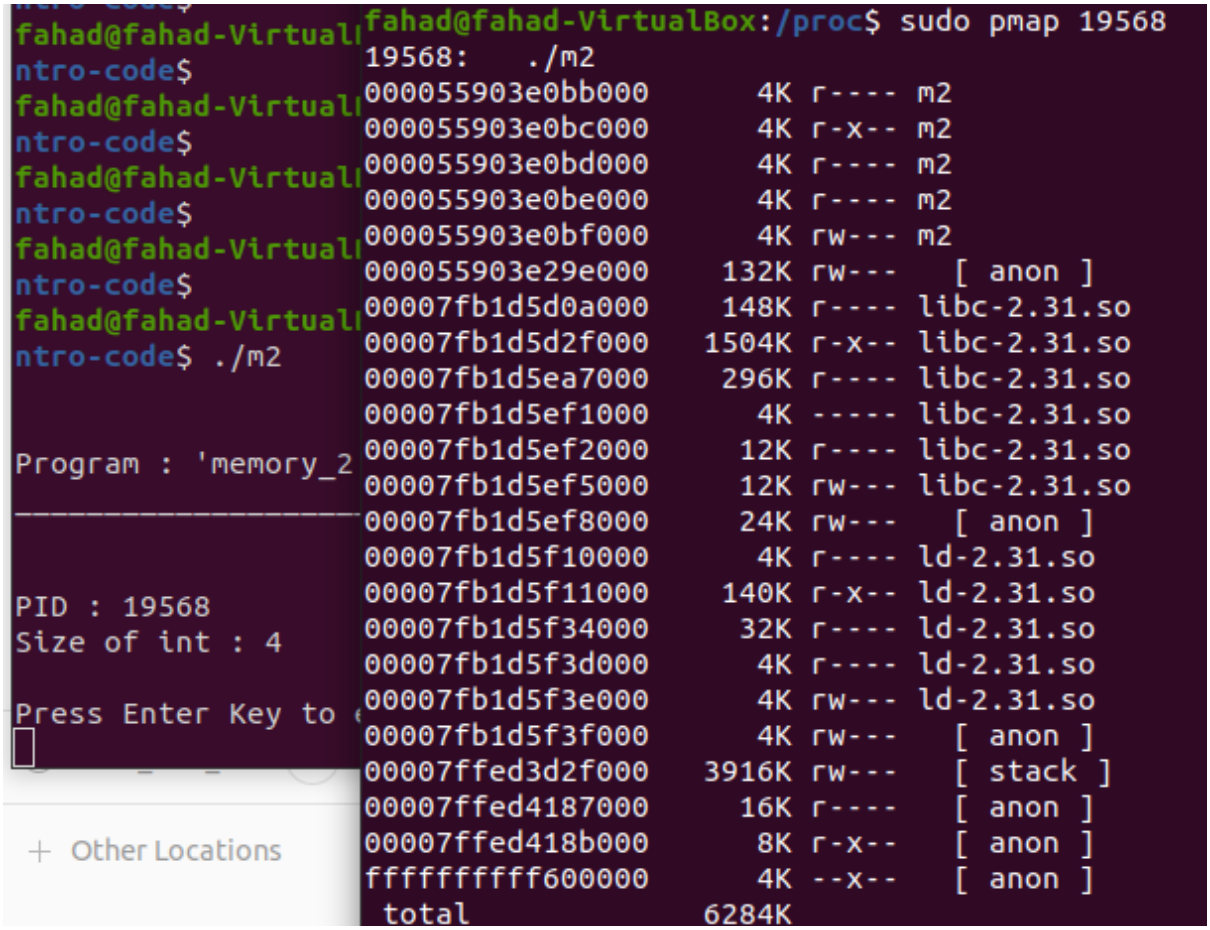
Ps is implemented by bash code itself.

Question 4

memory1.c



memory2.c


fahad@fahad-Virtuali
ntro-code$
fahad@fahad-Virtuali
ntro-code$
fahad@fahad-Virtuali
ntro-code$
fahad@fahad-Virtuali
ntro-code$
fahad@fahad-Virtuali
ntro-code$ ./m2
Program : 'memory_2
PID : 19568
Size of int : 4
Press Enter Key to
+ Other Locations
fahad@fahad-VirtualBox:/proc$ sudo pmap 19568
19568: ./m2
000055903e0bb000 4K r---- m2
000055903e0bc000 4K r-x-- m2
000055903e0bd000 4K r---- m2
000055903e0be000 4K r---- m2
000055903e0bf000 4K rw--- m2
000055903e29e000 132K rw--- [anon]
00007fb1d5d0a000 148K r---- libc-2.31.so
00007fb1d5d2f000 1504K r-x-- libc-2.31.so
00007fb1d5ea7000 296K r---- libc-2.31.so
00007fb1d5ef1000 4K ----- libc-2.31.so
00007fb1d5ef2000 12K r---- libc-2.31.so
00007fb1d5ef5000 12K rw--- libc-2.31.so
00007fb1d5ef8000 24K rw--- [anon]
00007fb1d5f10000 4K r---- ld-2.31.so
00007fb1d5f11000 140K r-x-- ld-2.31.so
00007fb1d5f34000 32K r---- ld-2.31.so
00007fb1d5f3d000 4K r---- ld-2.31.so
00007fb1d5f3e000 4K rw--- ld-2.31.so
00007fb1d5f3f000 4K rw--- [anon]
00007ffed3d2f000 3916K rw--- [stack]
00007ffed4187000 16K r---- [anon]
00007ffed418b000 8K r-x-- [anon]
ffffffffff600000 4K --x-- [anon]
total 6284K