EE-431L

Operating Systems

Lab

Submitted To

Sir Nauman Ahmad

Spring 2021

Lab Report # 1

Submitted By

Zafar Iqbal	2017-EE-69
Khurram Shehzad	2017-EE-77

Date Submitted: March 15, 2021

Department of Electrical Engineering

University of Engineering and Technology Lahore

LAB: INTRODUCTION

Question # 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.

```
khurram@khurram-VirtualBox:~$ more /proc/cpuinfo
```

```
khurram@khurram-VirtualBox:~$ lscpu
```

Processor: The processor is an electronic circuit inside the computer that carries out instruction to perform arithmetic, logical, control and input/output operations.

Core: The core refers to an execution unit inside the CPU that receives and executes instructions.

b. How many cores does your machine have?

Our machine has 4 cores.

```
cpu cores       : 4
```

c. How many processors does your machine have?

Our machine has 4 processors.

```
CPU(s):                          4
```

d. What is the frequency of each processor?

The frequency of each processor is 1800.002 MHz.

```
cpu MHz         : 1800.002
```

e. How much physical memory does your system have?

```
khurram@khurram-VirtualBox:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           1986         675         460           7         849        1144
Swap:          1450           0        1450
```

Total available physical memory is 1986 MB.

f. How much of this memory is free?

Free memory is 460 MB.

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

```
khurram@khurram-VirtualBox:~$ vmstat -f
         1859 forks
```

By running the above command, the total number of forks are 1859.

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

```
khurram@khurram-VirtualBox:~$ more /proc/stat
```

```
ctxt 391427
```

No. of context switches performed by the system are 391427.

Question # 2

In this question, we will understand how to monitor the status of a running process using the top command.

a. What is the PID of the process running the cpu command?

```
khurram@khurram-VirtualBox:~$ cd Desktop
khurram@khurram-VirtualBox:~/Desktop$ cd intro-code
khurram@khurram-VirtualBox:~/Desktop/intro-code$ gcc cpu.c -o cpu
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./cpu
```

```
khurram@khurram-VirtualBox:~$ ps -r
    PID TTY      STAT   TIME COMMAND
   1881 pts/0    R+     0:33 ./cpu
```

The PID of the process running the cpu command is 1881.

b. How much CPU and memory does this process consume?

```
khurram@khurram-VirtualBox:~$ top
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1881	khurram	20	0	2364	588	520	R	100.0	0.0	2:35.13	cpu

% CPU = 100 %
Current memory consumed is 588 KB.
Virtual memory is 2364 KB.
Memory percentage is 0.

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

Question # 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. Compiling the cpu-print.c

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ gcc cpu-print.c -o cpu-print
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./cpu-print
khurram@khurram-VirtualBox:~$ ps -r
    PID TTY      STAT   TIME COMMAND
   1948 pts/0    R+     0:01 ./cpu-print
```

The PID is 1948.

b. By running the following commands,

```
khurram@khurram-VirtualBox:~$ pstree -p -s 1948
systemd(1)---systemd(945)---gnome-terminal-(1676)---bash(1754)---cpu-print(1948)
```

PID of parent process is 1754.

PID of ancestors:

systemd: 1

systemd: 945

gnome-terminal: 1676

bash: 1754

c. By running the command,

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./cpu-print > /tmp/tmp.txt &
khurram@khurram-VirtualBox:~$ ps -r
    PID TTY      STAT   TIME COMMAND
   1956 pts/0    R      0:51 ./cpu-print
```

```
khurram@khurram-VirtualBox:~$ ls -l /proc/1956/fd
total 0
lrwx------ 1 khurram khurram 64 0 19:28 13   ماج -> /dev/pts/0
l-wx------ 1 khurram khurram 64 1 19:28 13   ماج -> /tmp/tmp.txt
lrwx------ 1 khurram khurram 64 2 19:28 13   ماج -> /dev/pts/0
```

(input) 0 is pointing to /dev/pts/0

(output) 1 is pointing to /tmp/tmp.txt

(error) 2 is pointing to /dev/pts/0

From the above observation, the shell performs I/O redirection by taking the file descriptor from the parent process for the child process and then with open command changes file descriptors.

d. By running the commands,

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./cpu-print | grep hello &
khurram@khurram-VirtualBox:~$ ps -r
 1967 pts/0    R      0:28 ./cpu-print
khurram@khurram-VirtualBox:~$ ls -l /proc/1967/fd
total 0
lrwx------ 1 khurram khurram 64 0 19:34 13   ماج -> /dev/pts/0
l-wx------ 1 khurram khurram 64 1 19:34 13   ماج -> 'pipe:[39024]'
lrwx------ 1 khurram khurram 64 2 19:34 13   ماج -> /dev/pts/0
```

(input) 0 is pointing to /dev/pts/0

(output) 1 is pointing to 'pipe:[39024]'

(error) 2 is pointing to /dev/pts/0

From the above observation, the shell implements pipes by linking the output of the parent process with the input of the child process.

e. **ls** and **ps** are executed by bash code itself. **history** and **cd** are shell built-in functions.

```
khurram@khurram-VirtualBox:~$ type -a cd
cd is a shell builtin
```

```
khurram@khurram-VirtualBox:~$ type -a ls
ls is aliased to `ls --color=auto'
ls is /usr/bin/ls
ls is /bin/ls
khurram@khurram-VirtualBox:~$ type -a history
history is a shell builtin
khurram@khurram-VirtualBox:~$ type -a ps
ps is /usr/bin/ps
ps is /bin/ps
```

Question # 4

After compiling and running the codes we get the following output.

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ gcc memory1.c -o memory1
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./memory1
```

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ gcc memory2.c -o memory2
```

```
khurram@khurram-VirtualBox:~/Desktop/intro-code$ ./memory2
```

```
khurram@khurram-VirtualBox:~$ top -p 1999
```

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1999 khurram   20   0    6280   4928    948 S   0.0   0.1   0:00.00 memory1
```

Virtual memory of memory1.c is 6280.

Physical resident memory of memory1.c is 4928.

```
khurram@khurram-VirtualBox:~$ top -p 2002
```

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2002 khurram   20   0    6284   4940    960 S   0.0   0.1   0:00.00 memory2
```

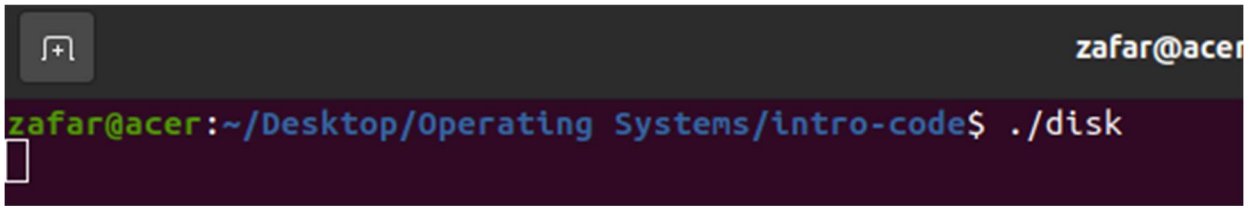
Virtual memory of memory1.c is 6284.

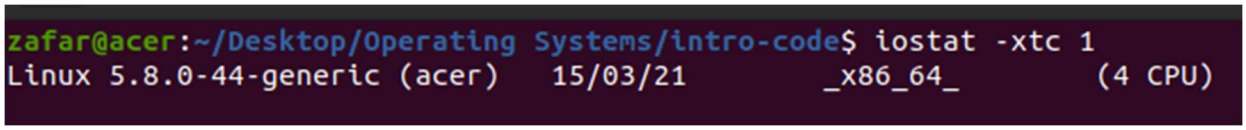
Physical resident memory of memory1.c is 4940.

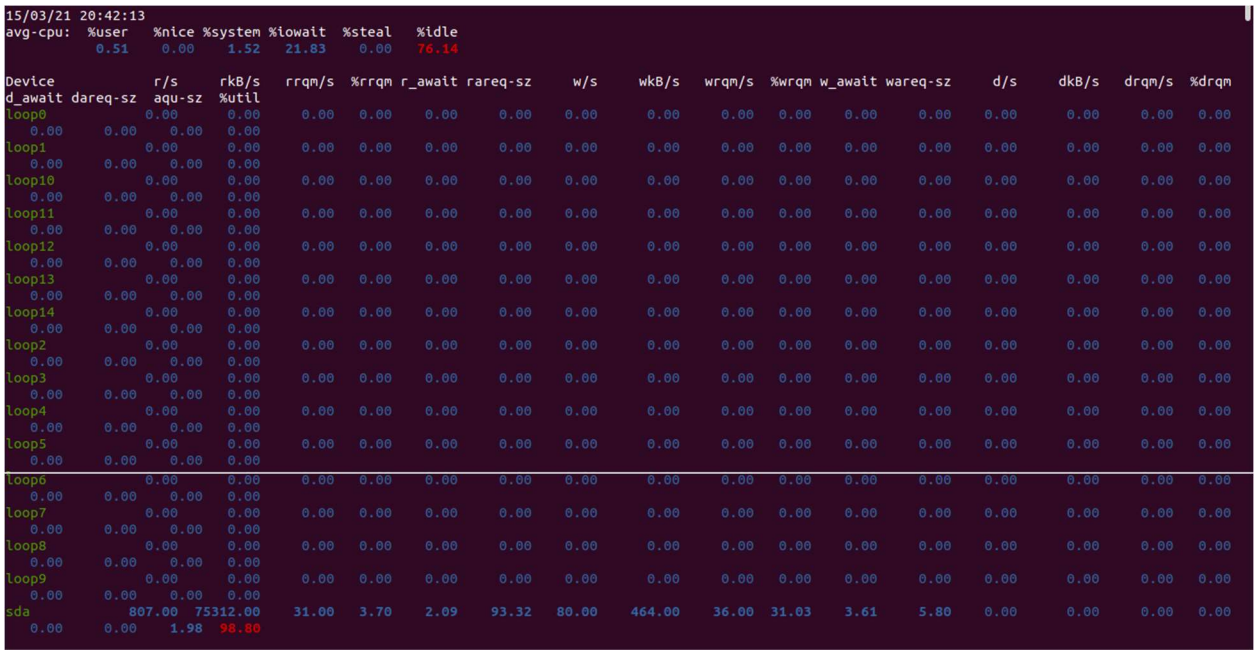
The virtual memory size of both the processes is almost same because the size of the array is same in both the cases which can be observed from the code. The physical memory size of both the processes is also almost identical for both the processes.

Question # 5

First, we compiled disk and disk1 using the gcc command. After that we made 5000 copies of foo.pdf by running the make-copies.sh in the linux terminal. Now by running disk.c and using the iostat command, we got the following output,





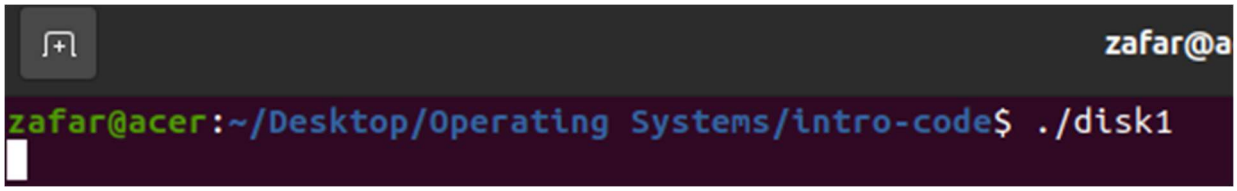


```
15/03/21 20:42:13
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.51    0.00    1.52   21.83    0.00   76.14

Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/s   drqm/s  %drqm
d_await dareq-sz  aqu-sz  %util
loop0            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop1            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop10           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop11           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop12           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop13           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop14           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop2            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop3            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop4            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop5            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop6            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop7            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop8            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop9            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
sda            807.00  75312.00    31.00   3.70    2.09    93.32   80.00    464.00    36.00  31.03    3.61     5.80    0.00      0.00     0.00   0.00
   0.00     0.00    1.98  98.80
```

Form above observation we see that when disk.c is running the disk utilization is close to 100 %.

Now by running disk1.c and using the iostat command, we got the following output,



```
zafar@acer:~/Desktop/Operating Systems/intro-code$ iostat -xtc 1
Linux 5.8.0-44-generic (acer)   15/03/21        _x86_64_        (4 CPU)
```

```
15/03/21 20:51:05
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          13.57    0.00   12.31    0.00    0.00   74.12

Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz     d/s     dkB/s   drqm/s  %drqm
d_await dareq-sz  aqu-sz  %util
loop0            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop1            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop10           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop11           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop12           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop13           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop14           0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop2            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop3            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop4            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop5            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop6            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop7            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop8            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
loop9            0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
sda              0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00    0.00     0.00    0.00      0.00     0.00   0.00
   0.00     0.00    0.00   0.00
```

Form above observation we see that when disk1.c is running the disk utilization is 0.

The difference in disk utilization is because disk.c is reading randomly selected different files which can be observed from the code and so many files cannot be placed in the cache that is why these files have to be read again and again from the disk, while disk1.c is only reading 1 file which can also be observed from its code. This 1 file is already in the cache and is being read again and again, that is why the disk utilization is 0 in case of disk1.c

The following code is being used to clear disk buffer cache,

```
zafar@acer: ~/Desktop/Operating Systems/intro-code
zafar@acer:~/Desktop/Operating Systems/intro-code$ sudo sh -c 'echo 3 > /proc/sys/vm/drop_caches'
```