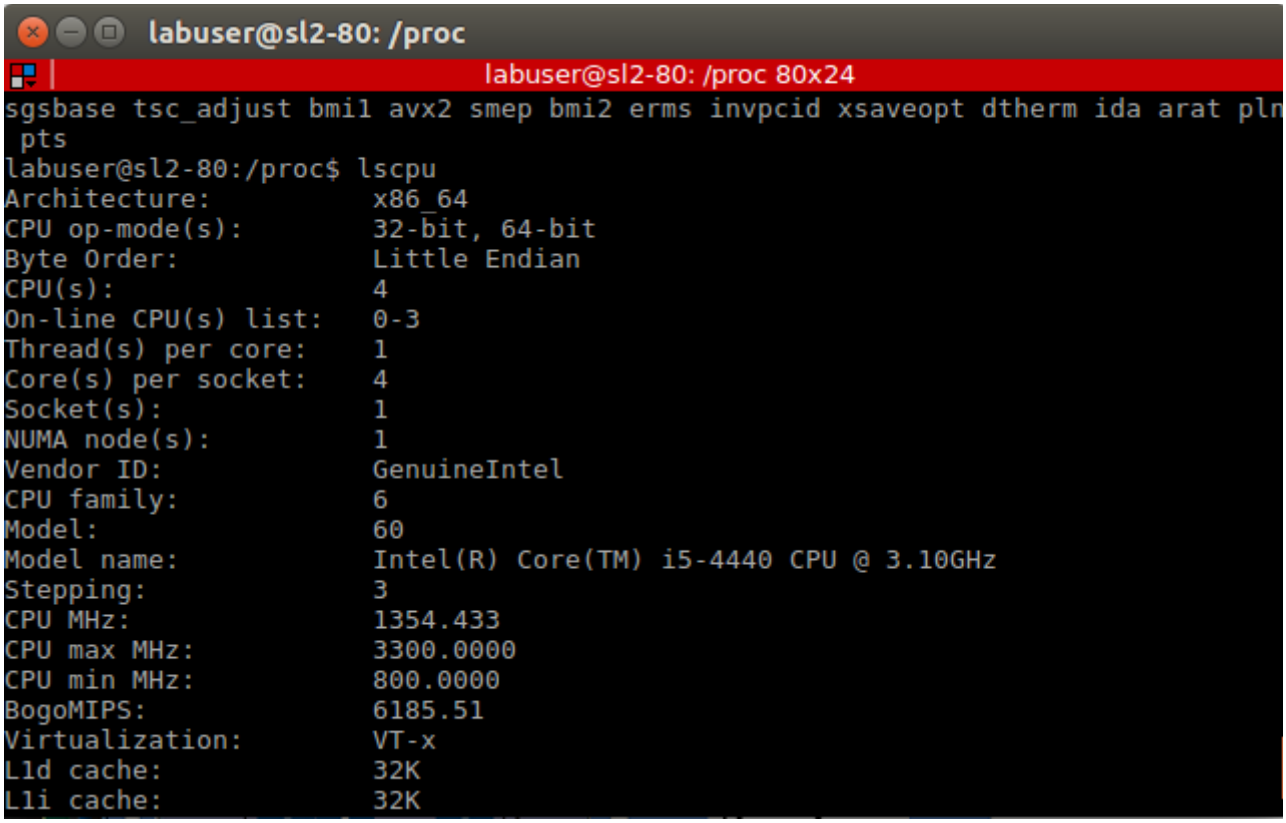
Lab 1
Roll number: 150050061

1. (a) I went to /proc/cpuinfo and wrote the command ‘lscpu’ to find out the information about the CPU.
Number of sockets = 1
Number of cores = 4
Number of threads per core = 1
Number of CPUs = No. of cores*No. of threads per core = 4*1 = 4



```
labuser@sl2-80: /proc
labuser@sl2-80: /proc 80x24
sgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln
 pts
labuser@sl2-80:/proc$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 60
Model name:            Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz
Stepping:              3
CPU MHz:               1354.433
CPU max MHz:           3300.0000
CPU min MHz:           800.0000
BogoMIPS:              6185.51
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
```

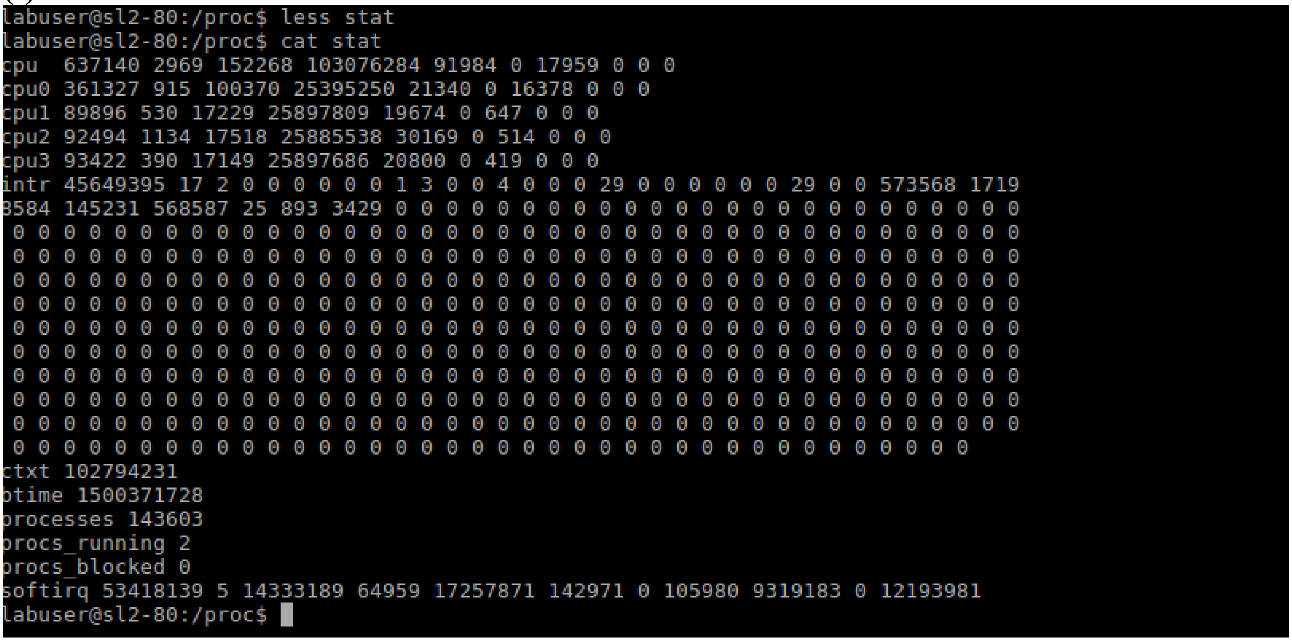
(b) I did ‘less /proc/cpuinfo’ to find out the current CPU frequency.
CPU0 freq: 3082.804 Mhz
CPU1 freq: 1592.261 Mhz
CPU2 freq: 2493.925 Mhz
CPU3 freq: 2990.046 Mhz
The text file is too big, I have saved it in **dump1b.txt**

(c) Did a ‘less /proc/meminfo’ to find out the total memory. Total memory is 7825944 kB.
Dump saved at **dump1c.txt**

(d) The above command also had the free and available memory details. Check **dump1c.txt**
MemFree: 4252596 kB
MemAvailable: 6064192 kB
The difference is that **MemFree** gives the physical amount of RAM that is free, but **MemAvailable** is the actual amount of memory that can be used by applications and processes.

(e) Did ‘cat stat’ inside /proc and saw value of ‘processes’. Here is the screenshot (next page)
Number of forks: 143603

(f) Same command and saw the value of ‘ctxt’. Number of context switches: 102794231



```
labuser@sl2-80:/proc$ less stat
labuser@sl2-80:/proc$ cat stat
cpu  637140 2969 152268 103076284 91984 0 17959 0 0 0
cpu0 361327 915 100370 25395250 21340 0 16378 0 0 0
cpu1 89896 530 17229 25897809 19674 0 647 0 0 0
cpu2 92494 1134 17518 25885538 30169 0 514 0 0 0
cpu3 93422 390 17149 25897686 20800 0 419 0 0 0
intr 45649395 17 2 0 0 0 0 0 0 1 3 0 0 4 0 0 0 29 0 0 0 0 0 0 29 0 0 573568 1719
3584 145231 568587 25 893 3429 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 102794231
btime 1500371728
processes 143603
procs_running 2
procs_blocked 0
softirq 53418139 5 14333189 64959 17257871 142971 0 105980 9319183 0 12193981
labuser@sl2-80:/proc$ 
```

2. I copied the VmRSS and VmSize from /proc/[pid]/status (dumps are present as **1.txt, 2.txt, 3.txt, 4.txt**)

memory_1.c :
VmRSS:	 688 kB
VmSize:	8140 kB

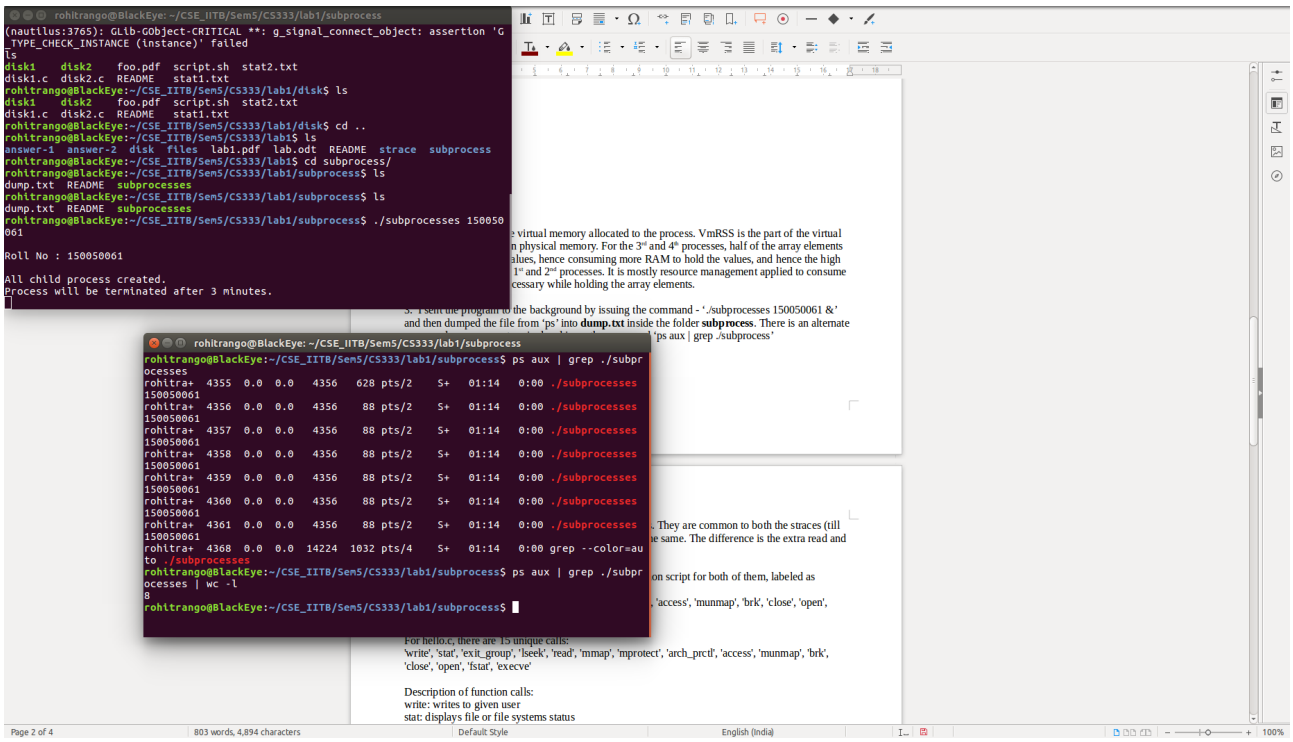
memory_2.c :
VmRSS:	 692 kB
VmSize:	12052 kB

memory_3.c :
VmRSS:	 4676 kB
VmSize:	8144 kB

memory_4.c :
VmRSS:	 6932 kB
VmSize:	12052 kB

VmSize is the size of the virtual memory allocated to the process. VmRSS is the part of the virtual memory that is present in physical memory. For the 3rd and 4th processes, half of the array elements are initialized to some values, hence consuming more RAM to hold the values, and hence the high VmRSS as compared to 1st and 2nd processes. It is mostly resource management applied to consume as much RAM that is necessary while holding the array elements.

3. I sent the program to the background by issuing the command - ‘./subprocesses 150050061 &’ and then dumped the file from ‘ps’ into **dump.txt** inside the folder **answer-3**. There is an alternate command, open a new terminal and issue the command ‘ps aux | grep ./subprocess’



4. Ran both and emptied the dump.
Note: Got some additional system calls to other libraries when I ran from my laptop.

(a) There are some ‘stat’ and ‘open’ calls to the libraries. They are common to both the straces (till line 38). The last line which makes an exit call is also the same. The difference is the extra read and write calls made by the hello.c file.

(b) There are 13 unique calls in empty.c file (ran a python script for both of them, labeled as script.py inside the folder **strace**) :
'stat', 'exit_group', 'read', 'mmap', 'mprotect', 'arch_prctl', 'access', 'munmap', 'brk', 'close', 'open', 'fstat', 'execve'

For hello.c, there are 15 unique calls:
'write', 'stat', 'exit_group', 'lseek', 'read', 'mmap', 'mprotect', 'arch_prctl', 'access', 'munmap', 'brk', 'close', 'open', 'fstat', 'execve'

Description of function calls:
write: writes to given user
stat: displays file or file systems status
exit_group: exits all threads in a process
lseek: repositions the read and write offset
read: reads from a file descriptor
mmap: maps/unmaps files or devices into memory
mprotect: sets protection for a memory region. If some program tries to violate the access rules, it gets a SIGSEGV signal
arch_prctl: sets architecture-specific process and thread states
access: checks whether a process can access the given file path
munmap: similar to mmap
brk: change the location of the program’s break
close: close a file descriptor

open: start the program on a virtual terminal
fstat: return the details of the file
execve: executes the program given its filename

5. First, I did an ‘strace’ and stored the dump in **dump.txt.** Now all the files passed in the open function are the one that the program opened. A simple python script present in the folder does the job. Here are the files it opened (**on my laptop**):

/usr/local/cuda-8.0/lib64/tls/x86_64/libc.so.6
/usr/local/cuda-8.0/lib64/tls/libc.so.6
/usr/local/cuda-8.0/lib64/x86_64/libc.so.6
/usr/local/cuda-8.0/lib64/libc.so.6
tls/x86_64/libc.so.6
tls/libc.so.6
x86_64/libc.so.6
libc.so.6
/etc/ld.so.cache
/lib/x86_64-linux-gnu/libc.so.6
/tmp/welocme to OS
/tmp/CS333
/tmp/CS347

6. Here are the block devices, their mountpoints, and the file system types of the block devices. I used the command ‘lsblk -f’ to display this information.

```
rohitrango@BlackEye:~$ lsblk -f
NAME    FSTYPE LABEL    UUID                                 MOUNTPOINT
sr0
sda
├─sda4  ntfs   OS       86E2AFCEE2AFC133
├─sda2  ntfs   Recovery 5E08AC9F08AC77A3
├─sda9  ntfs   Restore  01D1065277672B70
├─sda12 swap            f51dbb04-8348-4720-9f41-079762e3eb7a [SWAP]
├─sda7  ntfs   Assets   01D1714A68003E20
├─sda10 ntfs   Misc.    3A12E08A12E04D07
├─sda5  ntfs            949EB8379EB81428
├─sda3  ext4            9ef9fe09-b82c-4251-ae02-2975605c295f
├─sda1  vfat   SYSTEM   A0D9-2C43                            /boot/efi
├─sda8  ntfs   Study    01D10653C24837A0
├─sda11 ext4            42e14cb8-58a7-4f04-83fb-0c29c22df040 /
└─sda6  ntfs   Software 01D1714A65874EE0
rohitrango@BlackEye:~$
```

7. I did the following in a different terminal before running the program ./disk1 : **iostat -x 1 > stat1.txt**

The **%util** shows the disk utilization at every 1 second. There is ~100% disk utilization while the program is running, because it has to fetch every file. It took around 167.058476 seconds to run.

Before running ./disk2, I did **iostat -x 1 > stat2.txt** in another terminal. There is a maximum of 1.2% disk utilization since the file is already cached. It also took around 1.412285 seconds to run.

```
rohitrango@BlackEye:~/CSE_IITB/Sem5/CS333/lab1/disk$ rm stat.txt
rohitrango@BlackEye:~/CSE_IITB/Sem5/CS333/lab1/disk$ ./disk1

Total time take :  167.058476 seconds
rohitrango@BlackEye:~/CSE_IITB/Sem5/CS333/lab1/disk$ ./disk2

Total time take :  1.412285 seconds
rohitrango@BlackEye:~/CSE_IITB/Sem5/CS333/lab1/disk$ gnome-open ../lab
```

The executables and stat files are present inside the directory **answer-7**.