CS1217 - Spring 2023 - Homework 2

Bhumika Mittal, Saptarishi Dhanuka

Question 1

proc is a pseudo-filesystem which provides an interface to kernel data structures. Most of its files are read-only as it is just a virtual filesystem. It contains information about the processes that are currently running.

```
cs1217@cs1217-devel:~$ cd /proc
cs1217@cs1217-devel:/proc$ ls
1     107   127   1516  16    1688  1780  1841  1899  198   2140  2240  264   294   33    614   676   791   87   99          devices       ioports      locks         schedstat     timer_list
10    1082  1280  1528  1622  169   1788  1842  19    1993  2148  2242  27    295   337   617   677   8     88   acpi        diskstats     irq          mdstat        scsi          tty
100   109   13    1532  1634  1699  1792  1848  1900  1998  2150  23    28    296   37    618   682   80    89   asound      dma           kallsyms     meminfo       self          uptime
101   11    14    1537  1653  17    18    1849  1905  2     2172  2314  280   297   4     619   691   81    9    bootconfig  driver        kcore        misc          slabinfo      version
102   110   1434  1541  166   1701  1816  1854  1907  20    2173  2379  288   298   478   623   699   82    90   buddyinfo   dynamic_debug keys         modules       softirqs      version_signature
103   1110  15    1549  1669  1705  1818  1855  1918  2020  2182  238   289   3     481   631   7     83    92   bus         execdomains   key-users    mounts        stat          vmallocinfo
104   118   1500  1553  167   1737  1828  1856  1919  2046  2184  24    29    30    5     635   719   84    93   cgroups     fb            kmsg         mtrr          swaps         vmstat
105   12    1507  1560  1678  175   1829  1861  1924  2077  2186  2437  290   300   570   637   722   847   95   cmdline     filesystems   kpagecgroup  net           sys           zoneinfo
106   121   1508  1561  168   1759  1833  1863  1962  2081  2187  25    291   31    6     644   744   849   96   consoles    fs            kpagecount   pagetypeinfo  sysrq-trigger
1062  122   1514  1565  1683  1765  1834  1867  197   21    22    26    292   32    612   663   747   85    97   cpuinfo     interrupts    kpageflags   partitions    sysvipc
1065  1244  1515  1584  1684  1773  1836  1880  1971  2108  2212  263   293   329   613   672   749   86    98   crypto      iomem         loadavg      pressure      thread-self
cs1217@cs1217-devel:/proc$
```

Question 2

The following command-line arguments can be provided to get only the processes that belong to a specified user, say cs304:

(a) **top -U cs304**
top provides the dynamic real-time view of the processes that are *running currently*. The -U is used to sort the processes that belong to the user cs304.

(b) **ps -u cs304**
ps stands for process status. Like top, it provides the view of all the processes. -u is used for displaying the processes for the given effective user ID (which is like a virtual ID on the basis of which OS makes decisions).

(c) **ps -U cs304**
-U is used for displaying the processes for the given real user ID (real UID is the ID of the user which owns the process).

Question 3

(a) The core requirements of any shell are:

- User interface which allows the user to interact with the operating system either through textual commands in a Command-Line Interpreter or through direct manipulation of graphical elements in a GUI. In a CLI, it needs to support commands which could have many arguments
- Have functionality for Input/Output redirection from and to files and commands
- Have functionality to implement piping of output of a command as input of another command
- Have builtin commands which don't have to be exec'ed by the shell separately such as cd and echo in bash.
- Storing environment variables like $PATH and $HOME
- Support for traversing the file system by changing directories

(b) The OS mainly needs to provide support to the shell by having a system call interface which actually implements functionality related to fork(), exec(), opening and closing files, pipe(), managing processes, doing I/O and anything else that can only be done in the privileged kernel mode. It also needs to have the support for environment variables such as $PATH

(c) The following steps will need to be implemented to have a basic, working shell program:

- Check if the 3 main file descriptors associated with stdin, stdout and stderr are open
- Have a main loop which reads a line of input from the user
- Parse the user input into the command and its arguments
- Start a process by calling the fork() syscall which duplicates the process into a child process.
- If the return value of fork() is 0 then we are in the child process and we call the exec() syscall with an argument which is the path to an executable binary file. This transforms the child process into a process specified by the binary like ls, ps, etc.
- If the return value of fork() is nonzero then we wait for the child process to finish

Question 4

(a) **cpu**:
Using the time command in linux, we can observe that this program spends most of its time (around 80%) in the user mode whereas it spends the rest 20% time in the kernel mode. In the following screenshot, the cpu program was run for 1.9 seconds before it was terminated using SIGKILL.

```
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ time ./cpu
^C

real    0m1.990s
user    0m1.502s
sys     0m0.442s
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$
```

From the code, we can observe that the program spends most of its time doing calculations using CPU, hence it stays in the user mode. It goes to kernel mode initially while importing libraries, assigning values to the variables and over-writing the computed values.

(b) **cpu-print**:
Using the time command in linux, we can observe that this program spends most of its time (around 90%) in the kernel mode whereas it spends the rest 10% time in the user mode. In the following screenshot, the cpu-print program was run for 7.36 seconds before it was terminated using SIGKILL.

```
1676375446 sec, 860193 usec
1676375446 sec, 860194 usec
1676375446 sec, 860195 usec
1676375446 sec, 860197 usec
1676375446 sec, 860198 usec
1676375446 sec, 860199 usec
1676375446 sec, 860200 usec
1676375446 sec, 860202 usec
1676375446 sec, 860203 usec
1676375446 sec, 860204 usec
1676375446 sec, 860206 usec
1676375446 sec, 860207 usec
1676375446 sec, 860208 usec
1676375446 sec, 860210 usec
1676375446 sec, 860211 usec
167637544^C

real    0m7.366s
user    0m0.419s
sys     0m1.272s
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$
```

From the code, we can observe that the program spends most of its time doing two functions (gettimeofday() and printf()) which in turn call syscalls which are used to get the time of the day and then print respectively. The computation for incrementing the count value, etc is done in the user mode.

Question 5

(a) The shell program is **bash**.

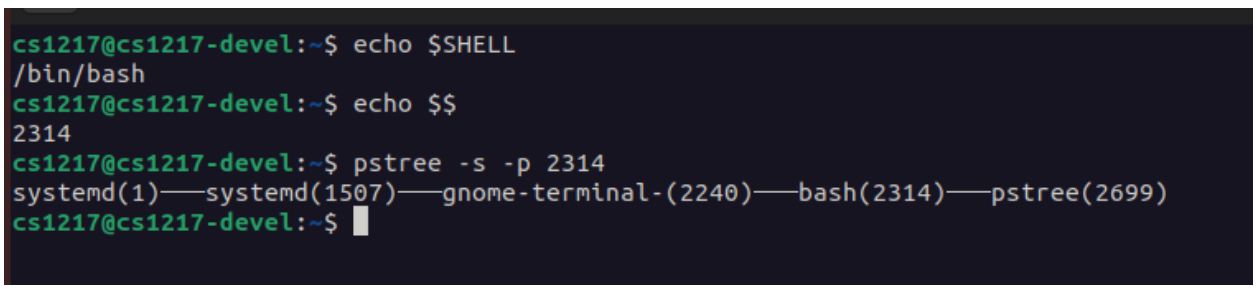
(b) It is maintained the variable **SHELL**.

(c) To print the same, we can use **echo $SHELL**.

(d) The pid of the shell could be printed using **echo $$**, which shows the pid of the current shell as 2314.

(e) The process tree is as follows:
systemd(1)−−−systemd(1507)−−−gnome-terminal-(2240)−−−bash(2314)−−−pstree(2699)

Here, systemd(1) is init, since it's pid is 1. The process tree can be obtained by using the following command: *pstree -s -p 2314*



-p flag is used to display the PIDs and -s shows parent processes of the specified process.

Question 6

The following system programs are simply **exec'ed** by the bash shell: **ls, ps**.
The system programs that are **implemented** by the bash code itself are: **cd, history**.

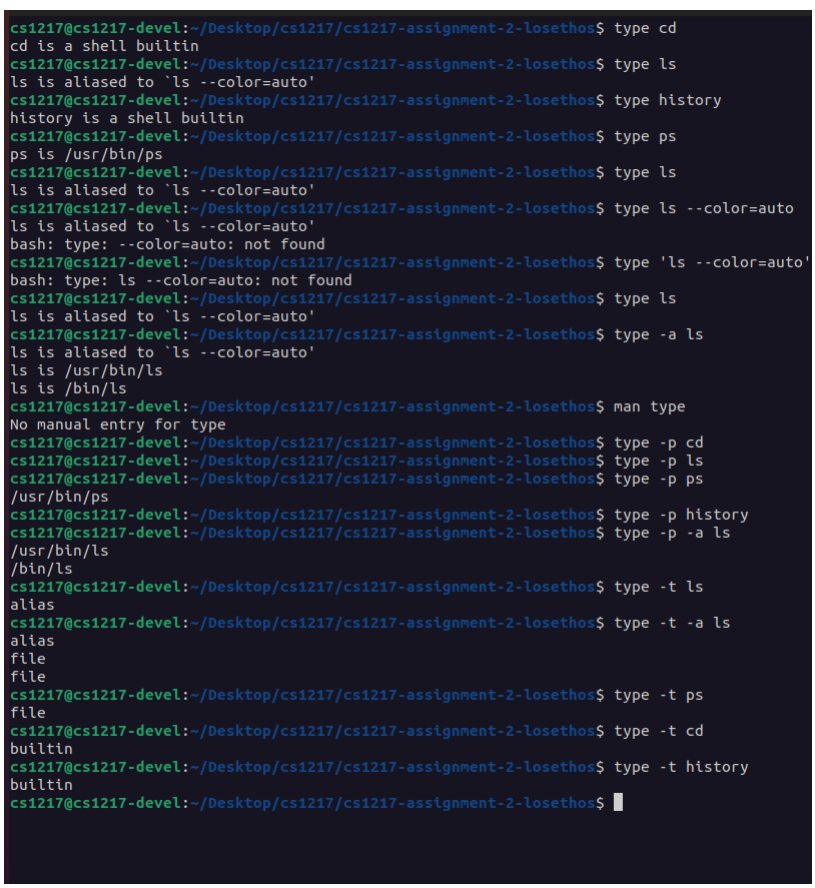
(a) **cd**: cd is implemented by the bash code itself. Using the type command we can see that the program is builtin and the strace command for the same indicated that it is not exec'ed by the bash shell.

(b) **ls**: ls is exec'ed by the bash shell. Using the type command with the -a flag (which deals with aliases) and strace clearly shows that ls is exec'ed by the bash code itself.

(c) **history**: history is implemented by the bash code itself. Using the type command we can see that the program is builtin and the strace command for the same indicated that it is not exec'ed by the bash shell.

(d) **ps**: ps is exec'ed by the bash shell. Using type command shows that ps is not a built-in command and strace clearly shows that ps command exec using the binary file located at the path of the command and then proceeds.

The following screenshots represents the output of type and strace command:



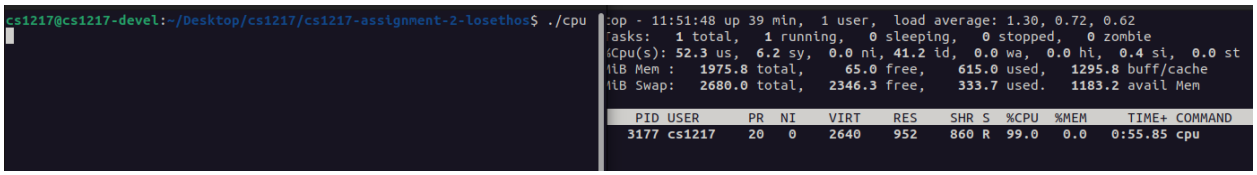
```
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type cd
cd is a shell builtin
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type ls
ls is aliased to `ls --color=auto'
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type history
history is a shell builtin
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type ps
ps is /usr/bin/ps
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type ls
ls is aliased to `ls --color=auto'
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type ls --color=auto
ls is aliased to `ls --color=auto'
bash: type: --color=auto: not found
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type 'ls --color=auto'
bash: type: ls --color=auto: not found
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type ls
ls is aliased to `ls --color=auto'
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -a ls
ls is aliased to `ls --color=auto'
ls is /usr/bin/ls
ls is /bin/ls
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ man type
No manual entry for type
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -p cd
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -p ls
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -p ps
/usr/bin/ps
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -p history
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -p -a ls
/usr/bin/ls
/bin/ls
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -t ls
alias
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -t -a ls
alias
file
file
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -t ps
file
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -t cd
builtin
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ type -t history
builtin
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$
```

```
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ strace -c ls
cpu  cpu.c  cpu-print  cpu-print.c  disk  disk1  disk1.c  disk.c  files  README.md
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 67.21    0.009276        9276         1           write
  8.53    0.001177          65        18           mmap
  4.74    0.000654          93         7           openat
  3.48    0.000480          53         9           close
  3.22    0.000444          55         8           newfstatat
  2.77    0.000383          63         6           mprotect
  2.06    0.000285          57         5           read
  1.28    0.000176          44         4           pread64
  0.99    0.000136          68         2           getdents64
  0.88    0.000122          61         2         2 statfs
  0.87    0.000120          60         2           ioctl
  0.78    0.000107          35         3           brk
  0.52    0.000072          72         1           munmap
  0.46    0.000063          31         2         2 access
  0.39    0.000054          54         1           set_tid_address
  0.38    0.000052          52         1           set_robust_list
  0.38    0.000052          52         1           prlimit64
  0.38    0.000052          52         1           rseq
  0.37    0.000051          51         1           getrandom
  0.33    0.000046          23         2         1 arch_prctl
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00    0.013802         176        78         5 total
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ strace -c cd
strace: Can't stat 'cd': No such file or directory
```

```
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ strace -c ps
    PID TTY          TIME CMD
   2385 pts/0    00:00:00 bash
   4503 pts/0    00:00:00 strace
   4506 pts/0    00:00:00 ps
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 44.37    0.004509          10       428           read
 19.46    0.001978           4       434         4 openat
 13.17    0.001338           3       430           close
 11.05    0.001123           4       229         3 newfstatat
  3.69    0.000375         187         2           getdents64
  3.14    0.000319           6        50           mmap
  1.98    0.000201          50         4           pread64
  1.44    0.000146           9        15           mprotect
  0.69    0.000070          17         4           write
  0.46    0.000047           2        22           rt_sigaction
  0.16    0.000016          16         1           munmap
  0.08    0.000008           1         6         4 prctl
  0.06    0.000006           3         2           ioctl
  0.05    0.000005           1         3           brk
  0.03    0.000003           1         2           lseek
  0.03    0.000003           1         2         1 arch_prctl
  0.03    0.000003           3         1           getrandom
  0.02    0.000002           2         1           geteuid
  0.02    0.000002           2         1           futex
  0.02    0.000002           2         1           set_tid_address
  0.02    0.000002           2         1           set_robust_list
  0.02    0.000002           2         1           prlimit64
  0.02    0.000002           2         1           rseq
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00    0.010162           6      1643        13 total
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$ strace -c history
strace: Can't stat 'history': No such file or directory
cs1217@cs1217-devel:~/Desktop/cs1217/cs1217-assignment-2-losethos$
```

Question 7

(a) **cpu**: The bottleneck resource for the cpu.c program is the **CPU**.
We ran the top command and executed the cpu.c file simultaneously. The %CPU field came out as around 100% for the cpu executable after running the command as visible in the following screenshot:



After reading the code of cpu.c, we can see that it contains an infinite while loop which in turn contains a for loop that calculates factorials. This process is a very resource heavy task using ALU unit of CPU. It keeps on executing this intensive for loop forever without any breaks in between which leads it to maximise CPU usage.

(b) **cpu-print**: The bottleneck resource for the cpu-print.c program is the **I/O**.
We ran the vmstat command multiple times at different intervals after executing cpu-print, and we can observe that the number in bi column (blocks read from the disk) is reducing, as visible in the following screenshot. Also, we can observe that when we run top command and check the details about various processes, cpu-print is switching state between R and S, and gnome-terminal along with cpu-print is consuming most of the %CPU resource, thus CPU acts as a secondary bottleneck

After reading the code of cpu-print.c, we can see that the program is making a system call to get the time of the day and to print the same. Since, it is printing continuously on the screen, the I/O is blocked by the input it is getting from the system calls, making I/O as the bottleneck.

```
cs1217@cs1217-devel: ~/Desktop/cs1217/cs1217-assignment-2-losethos
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 454588  40084 750284    0    0  1536    58  373 2917  4 10 85  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 454588  40092 750548    0    0  1507    58  425 5346  5 11 84  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 454588  40092 750396    0    0  1485    58  464 7755  5 11 83  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0      0 454588  40100 750308    0    0  1473    58  480 8906  5 12 83  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 4  0      0 454600  40164 750384    0    0  1326    58  707 22311  5 15 79  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 3  0      0 452736  40484 758064    0    0  1053    57 1068 45027  7 23 69  1  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 3  0      0 452488  40692 758364    0    0   806    58 1483 21431  8 30 61  0  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 452488  40768 758076    0    0   746    57 1574 29801  8 32 59  0  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 452488  40768 758280    0    0   744    57 1578 30184  9 32 59  0  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0      0 452488  40932 758224    0    0   698    58 1655 36895  9 33 58  0  0
cs1217@cs1217-devel:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 452488  40940 758308    0    0   696    58 1658 37008  9 33 58  0  0
cs1217@cs1217-devel:~$ vmstat
```

(c) **disk**: The bottleneck resource for the disk.c program is the **Disk**.
We ran the atop command and executed the disk.c file simultaneously. We observe that the program keeps switching state (S) between R(running) and D(uninterrupted-sleep, mainly caused due to disk full). Also, the DSK field came out as around 115% for the disk executable after running the command as visible in the following screenshot:

```
cs1217@cs1217-devel: ~/Desktop/cs1217/cs1217-assignment-2-losethos   cs1217@cs1217-devel: ~/Desktop/cs1217/cs1217-assignment-2-losethos
TOP - cs1217-devel                          2023/02/14  19:05:41                                                    10s elapsed
RC | sys   7.78s | user   1.03s |              | #proc    208 | #trun      1 | #tslpi   425 | #tslpu    49 | #zombie    0 | clones     0 |              |              | #exit      0 |
PU | sys     61% | user     9% |              | irq      41% |              | idle     70% | wait     19% |              | steal     0% | guest     0% |              |              | curf 2.61GHz |
pu | sys     41% | user     4% |              | irq      34% |              | idle     14% | cpu001 w  7% |              | steal     0% | guest     0% |              |              | curf 2.61GHz |
pu | sys     22% | user     5% |              | irq       9% |              | idle     52% | cpu000 w 12% |              | steal     0% | guest     0% |              |              | curf 2.61GHz |
PL | avg1    2.06 |              | avg5    1.70 | avg15   1.18 |              |              | csw    19363 |              | intr   21310 |              |              | numcpu     2 |              |
EM | tot     1.9G | free   74.9M | cache   1.2G | dirty   0.0M | buff   12.6M | slab  141.4M | slrec  81.4M | shmem  16.6M | shrss   0.0M | shswp   0.0M |              |              | numnode    1 |
WP | tot     2.6G |              | free    2.2G | swcac  54.2M |              |              |              |              |              |              | vmcom   3.8G | vmlim   3.6G |              |
   | scan  664382 | steal 624630 | stall      0 | compact    0 | numamig    0 | migrate 14e3 |              |              |              |              | swin       0 | swout   4242 | oomkill    0 |
SI | cpusome   9% | memsome   4% | memfull   4% |              | iosome   22% | iofull   20% | cs   8/13/14 | ms     3/1/0 | mf     3/1/0 | is    17/6/2 |              | if    15/5/1 |              |
   |          sda | busy    115% | read   12376 | write    449 | discrd     0 | KiB/r    100 | KiB/w     37 |              | KiB/d      0 | MBr/s  121.5 | MBw/s    1.7 | avq     1.40 | avio 0.76 ms |

  PID        SYSCPU        USRCPU        RDELAY        VGROW        RGROW        RUID         EUID         ST        EXC        THR        S        CPUNR        CPU        CMD        1/4
 5581        5.60s         0.05s         0.61s           0B           0B        cs1217       cs1217       --         -          1        D            1        66%        dlsk
 1662        0.36s         0.79s         0.88s        7.0M       -17.9M       cs1217       cs1217       --         -         14        S            0        13%        gnome-shell
   92        1.06s         0.00s         0.07s           0B           0B        root         root         --         -          1        S            0        12%        kswapd0
```

After reading the code of disk.c, we can see that the program is randomly opening any file from

the files folder, reading a block of data in 1024 bytes chunks by putting it into buffer and then discards the buffer. Since, there is an infinite loop which keeps opening random files, the disk space runs out quickly, thus becoming a bottleneck for the process.

(d) **disk1**: The bottleneck resource for the disk1.c program is the **CPU**.
We ran the top command and executed the disk1.c file simultaneously. The %CPU field came out as around 100% for the disk1 executable after running the command as visible in the following screenshot:

```
top - 19:30:42 up  8:18,  1 user,  load average: 0.38, 0.35, 0.74
Tasks: 203 total,   3 running, 200 sleeping,   0 stopped,   0 zombie
%Cpu(s): 21.8 us, 29.4 sy,  0.0 ni, 48.5 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
MiB Mem :   1975.8 total,     70.2 free,    603.6 used,   1302.0 buff/cache
MiB Swap:   2680.0 total,   2223.4 free,    456.6 used.   1184.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   5621 cs1217    20   0    2772    992    900 R  99.7   0.0   0:12.56 disk1
   1662 cs1217    20   0 4654156 222324  42060 S   2.0  11.0  13:22.57 gnome-shell
```

After reading the code of disk1.c, we can see that it contains an infinite while loop which opens and reads foo0.txt from the files subfolder. Like disk.c, it reads the data in the file in blocks of 1024 bytes. But it doesn't delete/remove the data it reads, which fills up the CPU quickly, hence making it a bottleneck for this program.