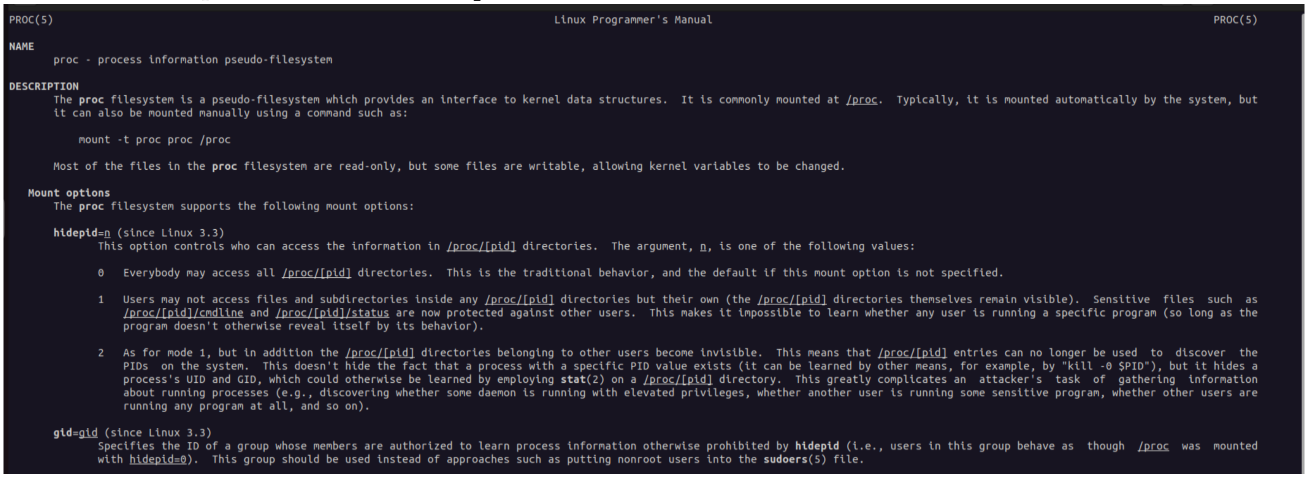
CS1217 - Spring 2023 - Homework 2

Gautam Ahuja, Nistha Singh

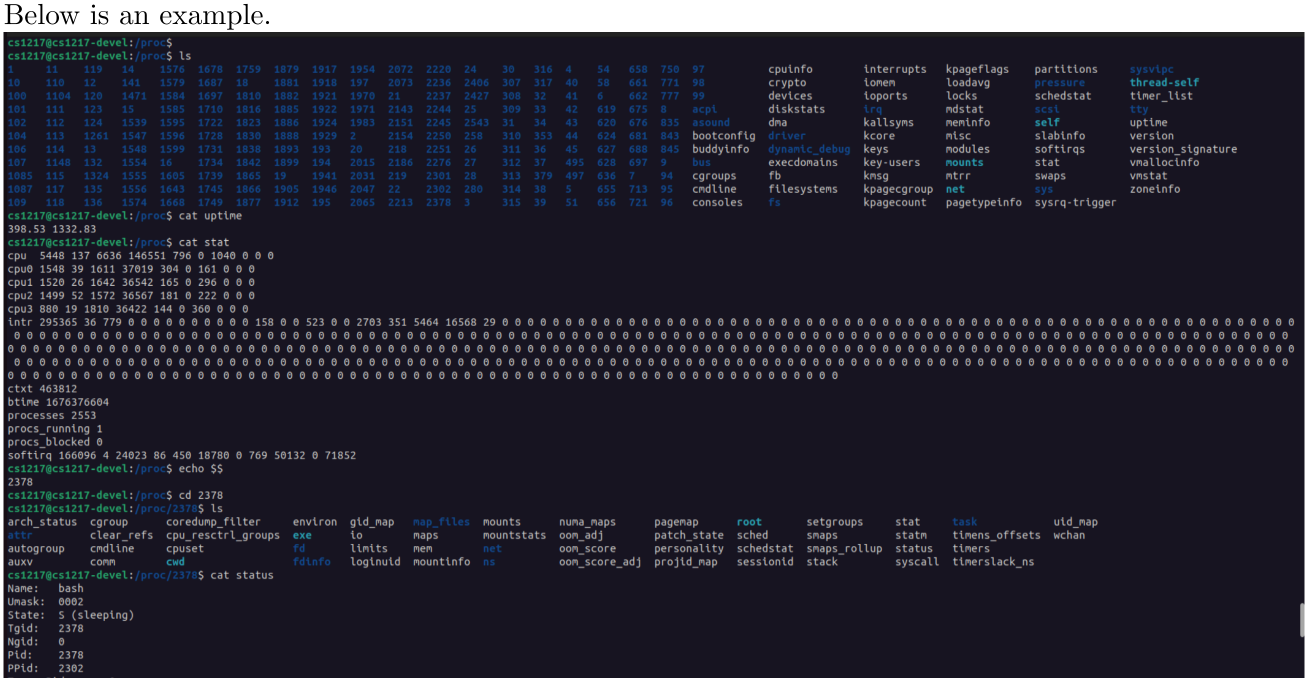
1. First Question

(a) Below is the implementation of `man proc`.



```
PROC(5)                                   Linux Programmer's Manual                                   PROC(5)

NAME
       proc - process information pseudo-filesystem

DESCRIPTION
       The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures.  It is commonly mounted at /proc.  Typically, it is mounted automatically by the system, but
       it can also be mounted manually using a command such as:

           mount -t proc proc /proc

       Most of the files in the proc filesystem are read-only, but some files are writable, allowing kernel variables to be changed.

   Mount options
       The proc filesystem supports the following mount options:

       hidepid=n (since Linux 3.3)
              This option controls who can access the information in /proc/[pid] directories.  The argument, n, is one of the following values:

              0   Everybody may access all /proc/[pid] directories.  This is the traditional behavior, and the default if this mount option is not specified.

              1   Users may not access files and subdirectories inside any /proc/[pid] directories but their own (the /proc/[pid] directories themselves remain visible).  Sensitive  files  such  as
                  /proc/[pid]/cmdline and /proc/[pid]/status are now protected against other users.  This makes it impossible to learn whether any user is running a specific program (so long as the
                  program doesn't otherwise reveal itself by its behavior).

              2   As for mode 1, but in addition the /proc/[pid] directories belonging to other users become invisible.  This means that /proc/[pid] entries can no longer be used  to  discover  the
                  PIDs  on the system.  This doesn't hide the fact that a process with a specific PID value exists (it can be learned by other means, for example, by "kill -0 $PID"), but it hides a
                  process's UID and GID, which could otherwise be learned by employing stat(2) on a /proc/[pid] directory.  This greatly complicates an  attacker's  task  of  gathering  information
                  about running processes (e.g., discovering whether some daemon is running with elevated privileges, whether another user is running some sensitive program, whether other users are
                  running any program at all, and so on).

       gid=gid (since Linux 3.3)
              Specifies the ID of a group whose members are authorized to learn process information otherwise prohibited by hidepid (i.e., users in this group behave as  though  /proc  was  mounted
              with hidepid=0).  This group should be used instead of approaches such as putting nonroot users into the sudoers(5) file.
```

(b) The \\`proc` file-system has contains many files which can be used to know more about system. Below is an example.



```
cs1217@cs1217-devel:/proc$
cs1217@cs1217-devel:/proc$ ls
1     11    119   14    1576  1678  1759  1879  1917  1954  2072  2220  24    30   316  4    54   658  750  97           cpuinfo        interrupts   kpageflags    partitions     sysvipc
10    110   12    141   1579  1687  18    1881  1918  197   2073  2236  2406  307  317  40   58   661  771  98           crypto         iomem        loadavg       pressure       thread-self
100   1104  120   1471  1584  1697  1810  1882  1921  1970  21    2237  2427  308  32   41   6    662  777  99           devices        ioports      locks         schedstat      timer_list
101   111   123   15    1585  1710  1816  1885  1922  1971  2143  2244  25    309  33   42   619  675  8    acpi         diskstats      irq          mdstat        scsi           tty
102   112   124   1539  1595  1722  1823  1886  1924  1983  2151  2245  2543  31   34   43   620  676  835  asound       dma            kallsyms     meminfo       self           uptime
104   113   1261  1547  1596  1728  1830  1888  1929  2     2154  2250  258   310  353  44   624  681  843  bootconfig   driver         kcore        misc          slabinfo       version
106   114   13    1548  1599  1731  1838  1893  193   20    218   2251  26    311  36   45   627  688  845  buddyinfo    dynamic_debug  keys         modules       softirqs       version_signature
107   1148  132   1554  16    1734  1842  1899  194   2015  2186  2276  27    312  37   495  628  697  9    bus          execdomains    key-users    mounts        stat           vmallocinfo
1085  115   1324  1555  1605  1739  1865  19    1941  2031  219   2301  28    313  379  497  636  7    94   cgroups      fb             kmsg         mtrr          swaps          vmstat
1087  117   135   1556  1643  1745  1866  1905  1946  2047  22    2302  280   314  38   5    655  713  95   cmdline      filesystems    kpagecgroup  net           sys            zoneinfo
109   118   136   1574  1668  1749  1877  1912  195   2065  2213  2378  3     315  39   51   656  721  96   consoles     fs             kpagecount   pagetypeinfo  sysrq-trigger
cs1217@cs1217-devel:/proc$ cat uptime
398.53 1332.83
cs1217@cs1217-devel:/proc$ cat stat
cpu  5448 137 6636 146551 796 0 1040 0 0 0
cpu0 1548 39 1611 37019 304 0 161 0 0 0
cpu1 1520 26 1642 36542 165 0 296 0 0 0
cpu2 1499 52 1572 36567 181 0 222 0 0 0
cpu3 880 19 1810 36422 144 0 360 0 0 0
intr 295365 36 779 0 0 0 0 0 0 0 0 0 0 158 0 0 523 0 0 2703 351 5464 16568 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 463812
btime 1676376604
processes 2553
procs_running 1
procs_blocked 0
softirq 166096 4 24023 86 450 18780 0 769 50132 0 71852
cs1217@cs1217-devel:/proc$ echo $$
2378
cs1217@cs1217-devel:/proc$ cd 2378
cs1217@cs1217-devel:/proc/2378$ ls
arch_status  cgroup      coredump_filter     environ  gid_map   map_files   mounts      numa_maps      pagemap      root        setgroups     stat     task            uid_map
attr         clear_refs  cpu_resctrl_groups  exe      io        maps        mountstats  oom_adj        patch_state  sched       smaps         statm    timens_offsets  wchan
autogroup    cmdline     cpuset              fd       limits    mem         net         oom_score      personality  schedstat   smaps_rollup  status   timers
auxv         comm        cwd                 fdinfo   loginuid  mountinfo   ns          oom_score_adj  projid_map   sessionid   stack         syscall  timerslack_ns
cs1217@cs1217-devel:/proc/2378$ cat status
Name:   bash
Umask:  0002
State:  S (sleeping)
Tgid:   2378
Ngid:   0
Pid:    2378
PPid:   2302
```

2. Second Question

The command `top` returns the status of all running processes (of kernel and all users) in real time. A demo of command:

```
top - 17:52:38 up 12 min,  1 user,  load average: 0.37, 0.26, 0.15
Tasks: 213 total,   1 running, 212 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.5 us,  1.3 sy,  0.0 ni, 97.0 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   3923.6 total,   2286.3 free,    837.8 used,    799.5 buff/cache
MiB Swap:   2680.0 total,   2680.0 free,      0.0 used.   2820.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1728 cs1217    20   0 5277520 406272 147636 S  17.3  10.1   2:28.39 gnome-shell
   2302 cs1217    20   0  572112  61832  45224 S   4.0   1.5   0:20.09 gnome-terminal-
   2245 cs1217    20   0  227512   2676   2316 S   1.7   0.1   0:09.72 VBoxClient
      9 root      20   0       0      0      0 I   0.7   0.0   0:08.32 kworker/u8:0-events_unbound
     27 root      rt   0       0      0      0 S   0.3   0.0   0:00.63 migration/2
    495 systemd+  20   0   14828   6088   5288 S   0.3   0.2   0:02.42 systemd-oomd
   1085 root      20   0  298324   1260   1128 S   0.3   0.0   0:01.53 VBoxDRMClient
   1734 root      20   0       0      0      0 I   0.3   0.0   0:03.14 kworker/0:3-events
   2237 cs1217    20   0  226996   2528   2168 S   0.3   0.1   0:01.54 VBoxClient
   2575 cs1217    20   0   21868   4240   3384 R   0.3   0.1   0:00.12 top
      1 root      20   0  166772  11892   8280 S   0.0   0.3   0:02.16 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthreadd
      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_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      7 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0-events
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     12 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     13 root      20   0       0      0      0 S   0.0   0.0   0:00.06 ksoftirqd/0
     14 root      20   0       0      0      0 I   0.0   0.0   0:01.80 rcu_sched
     15 root      rt   0       0      0      0 S   0.0   0.0   0:00.07 migration/0
     16 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     18 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     20 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     21 root      rt   0       0      0      0 S   0.0   0.0   0:00.64 migration/1
     22 root      20   0       0      0      0 S   0.0   0.0   0:00.12 ksoftirqd/1
     24 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
     25 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
     26 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/2
     28 root      20   0       0      0      0 S   0.0   0.0   0:00.08 ksoftirqd/2
     30 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H-events_highpri
     31 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/3
     32 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/3
     33 root      rt   0       0      0      0 S   0.0   0.0   0:00.64 migration/3
     34 root      20   0       0      0      0 S   0.0   0.0   0:00.10 ksoftirqd/3
     36 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/3:0H-kblockd
```

The command `ps` return the snapshot of the running processes in system in user mode for current user, where the command `ps au` returns the snapshot of all the running processes bu all different users.

```
cs1217@cs1217-devel:~$ ps
    PID TTY          TIME CMD
   2378 pts/0    00:00:00 bash
   2586 pts/0    00:00:00 ps
cs1217@cs1217-devel:~$
```

we can extract information of all running commands by using `grep` command. Pipe the output of `ps au` to `grep cs1217` and it will display all running tasks for user `cs1217`.

```
cs1217@cs1217-devel:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos  xv6-new.tar.xz  xv6-public
cs1217@cs1217-devel:~$ ps au | grep cs1217
cs1217      1595  0.0  0.1 171036  6192 tty2     Ssl+ 17:40   0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --session=ubuntu
cs1217      1605  0.0  0.3 231684 15336 tty2     Sl+  17:40   0:00 /usr/libexec/gnome-session-binary --session=ubuntu
cs1217      2378  0.0  0.1  20052  5640 pts/0    Ss   17:40   0:00 bash
cs1217      2775  0.0  0.0  21324  1556 pts/0    R+   18:09   0:00 ps au
cs1217      2776  0.0  0.0  17732  2296 pts/0    S+   18:09   0:00 grep --color=auto cs1217
cs1217@cs1217-devel:~$
```

3. Third Question

(a) A shell is a user program that reads commands from the user and executes them. It acts as an interface between the user and the kernel. The core requirements of any shell is to be able to smoothly perform its functionality which is:

1) Taking input from the user,
2) Parsing the entered commands and its arguments,
3) Return Errors if command is invalid,
4) Executing system calls, commands and libraries, and
5) Printing

(b) The shell's use of system calls illustrates how carefully they have been designed. In order to perform these system calls smoothly, there should be detailed provisions for privileged access (permissions and users). That is, a shell is a user program and can only access the functionality of a kernel (Which operates on privileged access) using system calls. The OS should also provide a mechanism for inter-process communication between processes and their children. OS should also help shell managing the memory, storage etc. of a process.

(c) Steps needed to implement a basic, working shell program:

1) A way to take user input so that the commands can be executed.
2) Forking and `exec()` (or other related system calls) a process
3) Execute the binary arguments
4) Giving privilege to parent process to implement the blocking wait or non-blocking wait.
5) If the blocking wait is implemented, then the parent will wait; and if the non-blocking wait is implemented, then it will start accepting more inputs and repeat from beginning.

4. Fourth Question

A program spends more time in kernel mode when it is making `syscall`(s) such as `fopen()`, `printf()`, `get()`, etc. The `cpu-print.c` program has `gettimeofday()`, `printf()` instruction, so intuitively it will spend more time in **kernel space**.

```
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ ls
README.md  cpu-print.c  cpu.c  disk.c  disk1.c
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ cat cpu.c
#include <unistd.h>
#include <stdio.h>

int main(int argc, char *argv[])

{
  unsigned int i,j;
  while(1)
    {
      j = 1;
      for(i = 1; i <= 10; i++)
        {
          j = j*i;
        }
    }
}


gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ cat cpu-print.c
#include <unistd.h>
#include <stdio.h>
#include <sys/time.h>

int main(int argc, char *argv[])

{
  unsigned int i;
  int count = 0;
  struct timeval tv;

  while(1)
    {
      for(i = 0; i < 1000000; i++)
        {
          gettimeofday(&tv, NULL);
          printf("%u sec, %u usec\n", tv.tv_sec, tv.tv_usec);
        }
      count++;
      printf("round %d complete\n", count);
    }
}
```

Similarly, the program `cpu.c` does not have `syscalls` and is executing user instructions. Hence it spends more time in **user space**.

The `time` command followed by a program, outputs the overall time spent by a program and also time spent in `user` and `kernel` mode, given by `real`, `user` and `sys` respectively.

The deduction that `cpu.c` spends more time in **user mode** and `cpu-print.c` spends more time in **kernel space** can also be confirmed by `time` command.

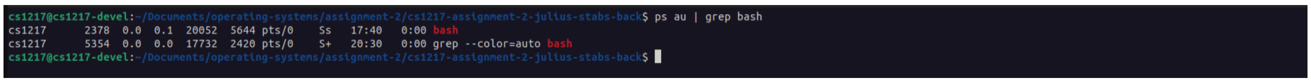
```
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ ls
README.md  cpu  cpu-print  cpu-print.c  cpu.c  disk.c  disk1.c
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ time ./cpu
^C
real    1m51.454s
user    1m50.952s
sys     0m0.500s

gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$
```

```
1676383567 sec, 448350 usec
1676383567 sec, 448351 usec
^C
real    0m33.953s
user    0m1.277s
sys     0m8.035s

gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$
```

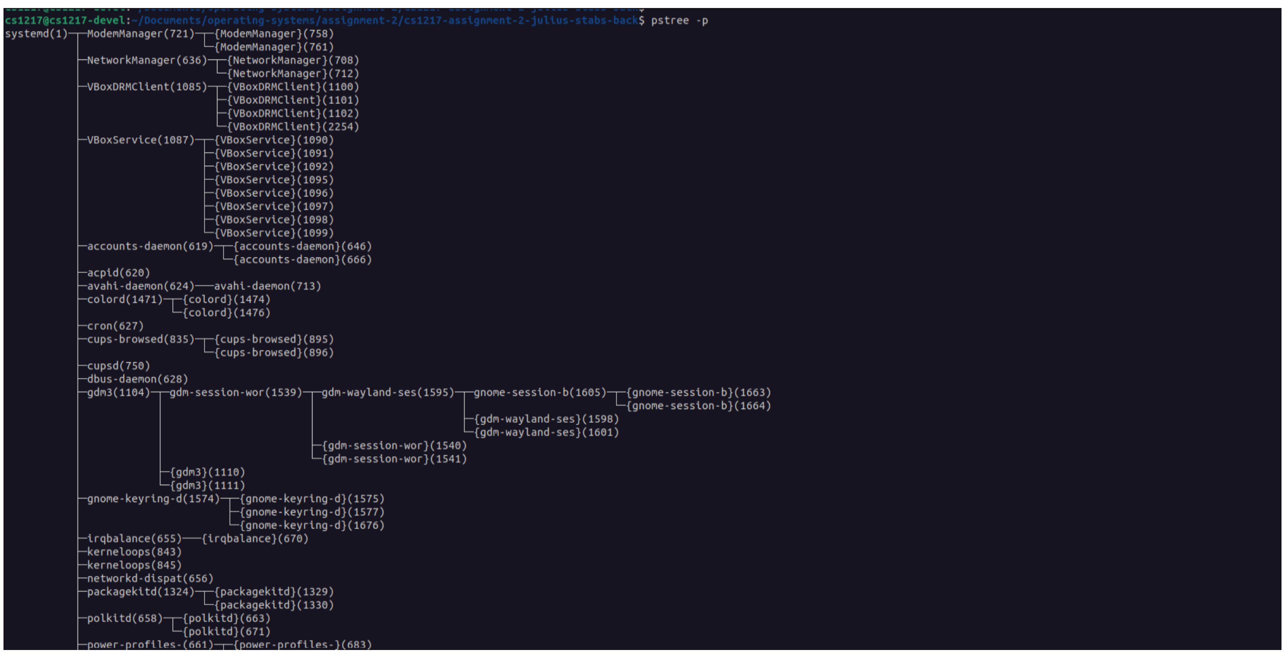
5. Fifth Question

As we all know (discussed in class), the name of the shell is stored in a variable called `SHELL`. It can be printed in command-line by using `echo $SHELL` on terminal. In this the shell is `bash`

The `PID` of the any process can be found by running the following command: `ps au | grep bash`. It return processes named `bash` and we can see the `PID` of the `bash`. Here, the `PID` is 2378.



To obtain the process tree, we run the command: `pstree -p`, this return a process tree along with the `PID`(s) of each process.



Here the process tree for `bash` is `systemd(1) -> systemd(1547) -> gnome-terminal(2302) -> bash(2378)`

6. Sixth Question

We can check if a command is built-in or `exec()`-ed using the `type` command. The command can be found inside `bash` documentation

```
gautam-ahuja@LAPTOP-FV7627LB:~$ type cd
cd is a shell builtin
gautam-ahuja@LAPTOP-FV7627LB:~$ type history
history is a shell builtin
gautam-ahuja@LAPTOP-FV7627LB:~$ type ls
ls is aliased to `ls --color=auto'
gautam-ahuja@LAPTOP-FV7627LB:~$ type ps
ps is /usr/bin/ps
gautam-ahuja@LAPTOP-FV7627LB:~$ 
```

Commands which are implemented by `bash`: `cd` and `history`

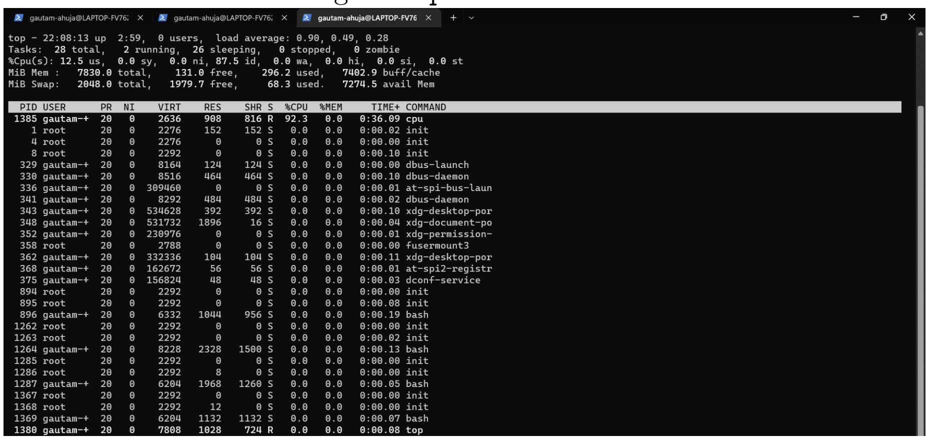
Commands which are `exec()`-ed: `ls` and `ps`

The `exec()` takes the binary ELF as input argument. One other way to figure out if the command is `exec()`-ed is see if the corresponding ELF binary is present in `/bin` directory or not.

7. Seventh Question

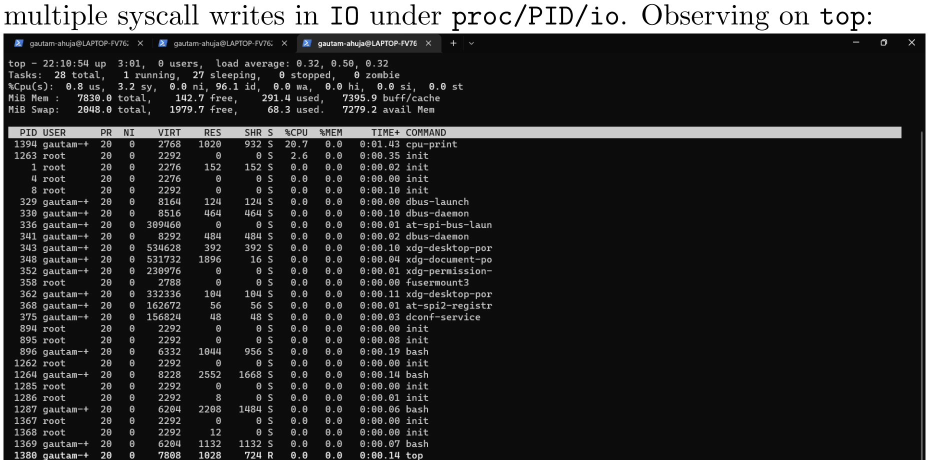
(a) For `cpu`,

1) **Bottleneck:** CPU Usage
2) **Reasoning Justification:** The code is multiplying `i*j`. We are only utilizing the CPU during multiplication up to 99%. Hence it is the bottleneck, as other processes wont be able to use the CPU. Observing on `top`:

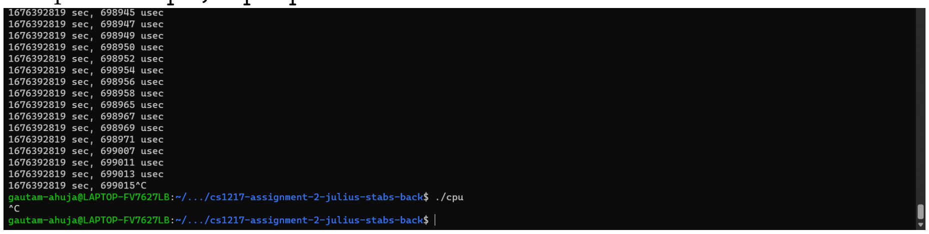


(b) For `cpu-print`,

1) **Bottleneck:** None, the program is not causing any bottleneck.
2) **Reasoning Justification:** Whenever each print happens, a system call happens to write on the terminal. Therefore, the control keeps going back and forth between the user and the kernel as there are multiple syscalls. So, it will depend on the implementation and if it turns out to be a system call then there might be a CPU bottleneck. We can also see multiple syscall writes in `IO` under `proc/PID/io`. Observing on `top`:



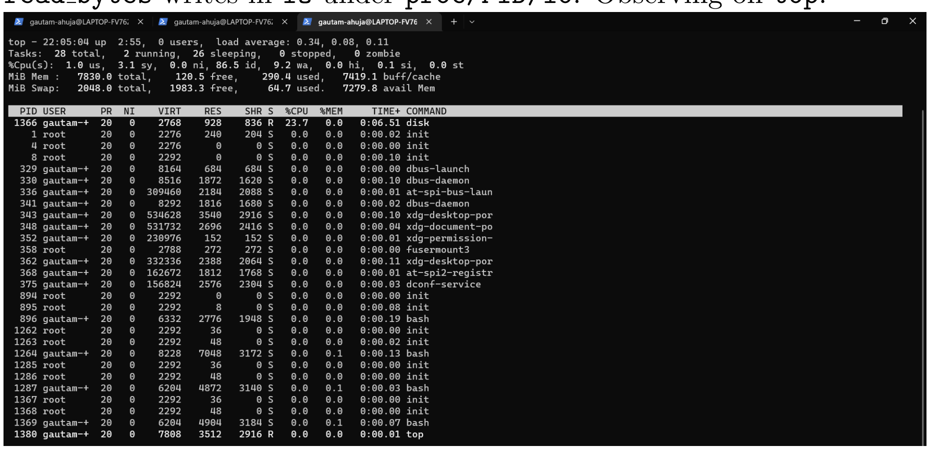
Output for `cpu, cpu-print`



For `disk`,

1) **Bottleneck:** STORAGE/DISK IO

2) **Reasoning Justification:** There is a lot of stroage/disk IO utilization as each time when a random file is selected, its content had to be read from the disk. also, there are 10,000 files, each of 2`MB`, which will result in 20`GB` size which won't fit into the memory. We can also see multiple `read_bytes` writes in `IO` under `proc/PID/io`. Observing on `top`:



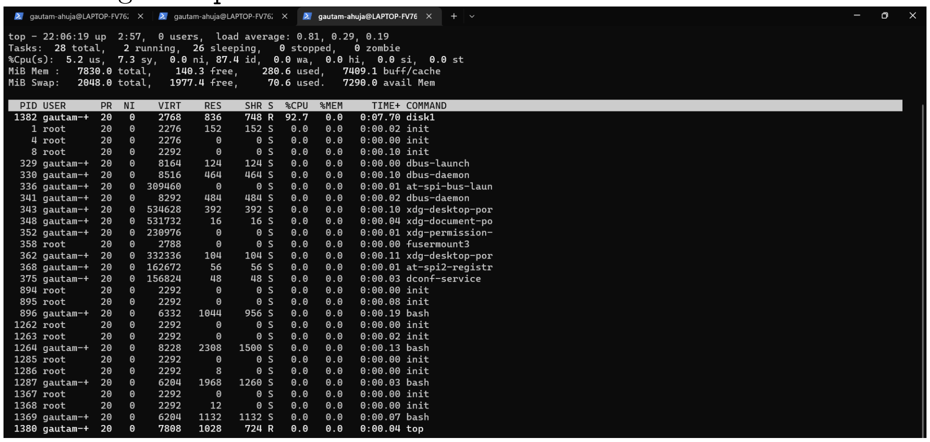
For `disk1`,

1) **Bottleneck:** No Bottleneck

2) **Reasoning Justification:** The CPU is only used 90%, and we cannot declare it a bottleneck as other processes can still use it. We are only accessing 1 file to read many times, as per the code. Hence there is no disk io bottleneck happening here. Hence no type pf bottleneck is

observed. We can also see that the contents in `IO` under `proc/PID/io` are not significant in numbers.

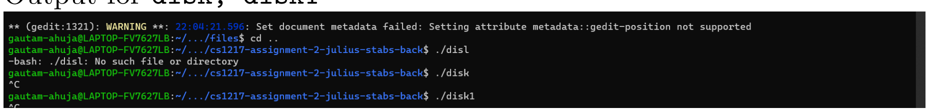
Observing on `top`:



```
top - 22:06:19 up  2:57,  0 users,  load average: 0.81, 0.29, 0.19
Tasks:  28 total,   2 running,  26 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.2 us,  7.3 sy,  0.0 ni, 87.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7830.0 total,    140.3 free,    280.6 used,   7409.1 buff/cache
MiB Swap:   2048.0 total,   1977.4 free,     70.6 used.   7290.0 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1382 gautam-+  20   0    2768    836    748 R  92.7   0.0   0:07.70 disk1
    1 root      20   0    2276    152    152 S   0.0   0.0   0:00.02 init
    4 root      20   0    2276      0      0 S   0.0   0.0   0:00.00 init
    8 root      20   0    2292      0      0 S   0.0   0.0   0:00.10 init
  329 gautam-+  20   0    8164    124    124 S   0.0   0.0   0:00.00 dbus-launch
  330 gautam-+  20   0    8516    464    464 S   0.0   0.0   0:00.10 dbus-daemon
  336 gautam-+  20   0  309460      0      0 S   0.0   0.0   0:00.01 at-spi-bus-laun
  341 gautam-+  20   0    8292    484    484 S   0.0   0.0   0:00.02 dbus-daemon
  343 gautam-+  20   0  534628    392    392 S   0.0   0.0   0:00.10 xdg-desktop-por
  348 gautam-+  20   0  531732     16     16 S   0.0   0.0   0:00.04 xdg-document-po
  352 gautam-+  20   0  230976      0      0 S   0.0   0.0   0:00.01 xdg-permission-
  358 root      20   0    2788      0      0 S   0.0   0.0   0:00.00 fusermount3
  362 gautam-+  20   0  332336    104    104 S   0.0   0.0   0:00.11 xdg-desktop-por
  368 gautam-+  20   0  162672     56     56 S   0.0   0.0   0:00.01 at-spi2-registr
  375 gautam-+  20   0  156824     48     48 S   0.0   0.0   0:00.03 dconf-service
  894 root      20   0    2292      0      0 S   0.0   0.0   0:00.00 init
  895 root      20   0    2292      0      0 S   0.0   0.0   0:00.08 init
  896 gautam-+  20   0    6332   1044    956 S   0.0   0.0   0:00.19 bash
 1262 root      20   0    2292      0      0 S   0.0   0.0   0:00.00 init
 1263 root      20   0    2292      0      0 S   0.0   0.0   0:00.02 init
 1264 gautam-+  20   0    8228   2308   1500 S   0.0   0.0   0:00.13 bash
 1285 root      20   0    2292      0      0 S   0.0   0.0   0:00.00 init
 1286 root      20   0    2292      8      0 S   0.0   0.0   0:00.00 init
 1287 gautam-+  20   0    6204   1968   1260 S   0.0   0.0   0:00.03 bash
 1367 root      20   0    2292      0      0 S   0.0   0.0   0:00.00 init
 1368 root      20   0    2292     12      0 S   0.0   0.0   0:00.00 init
 1369 gautam-+  20   0    6204   1132   1132 S   0.0   0.0   0:00.07 bash
 1380 gautam-+  20   0    7808   1028    724 R   0.0   0.0   0:00.04 top
```

```
gautam-ahuja@LAPTOP-FV7627LB:/proc$ cd 1382
gautam-ahuja@LAPTOP-FV7627LB:.../1382$ cat io
rchar: 181602058352
wchar: 0
syscr: 44423044
syscw: 0
read_bytes: 2854912
write_bytes: 0
cancelled_write_bytes: 0
gautam-ahuja@LAPTOP-FV7627LB:.../1382$
```

Output for `disk, disk1`



```
** (gedit:1321): WARNING **: 22:04:21.596: Set document metadata failed: Setting attribute metadata::gedit-position not supported
gautam-ahuja@LAPTOP-FV7627LB:~/.../files$ cd ..
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ ./disl
-bash: ./disl: No such file or directory
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ ./disk
^C
gautam-ahuja@LAPTOP-FV7627LB:~/.../cs1217-assignment-2-julius-stabs-back$ ./disk1
```