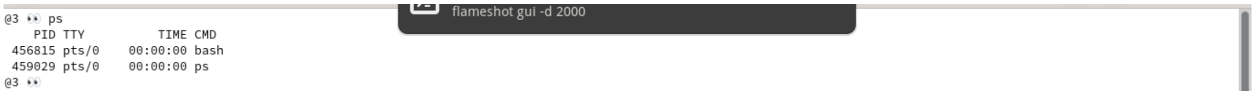
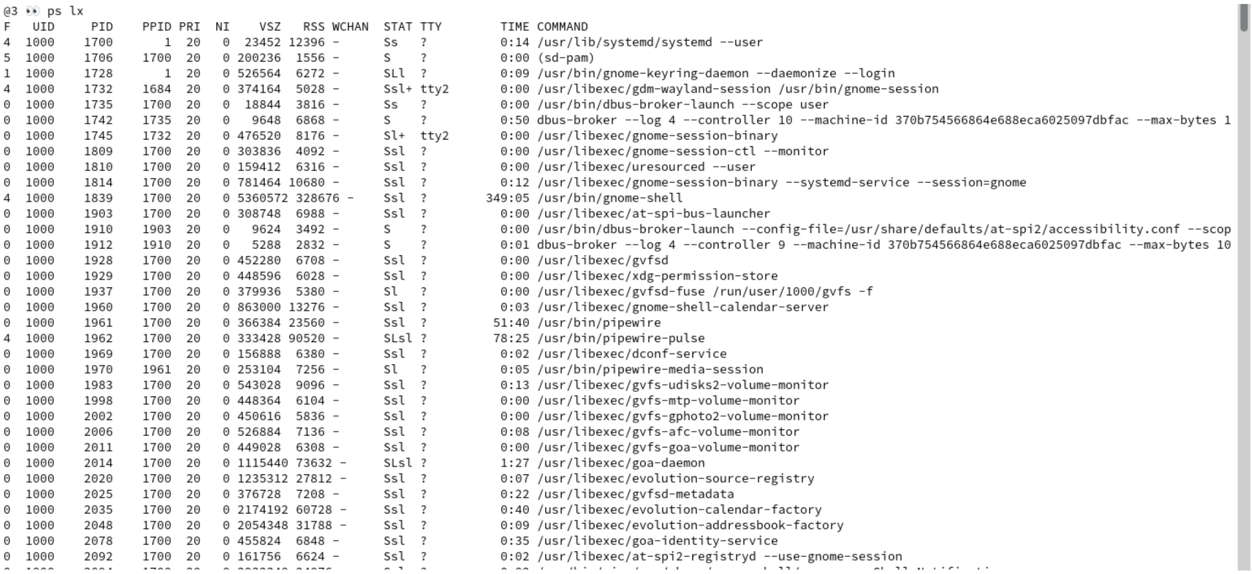
1. Use the ps, ps lx, ps tree and ps -aux command to display the process attributes.

ps output:



```
@3 👀 ps
    PID TTY          TIME CMD
 456815 pts/0    00:00:00 bash
 459029 pts/0    00:00:00 ps
@3 👀
```

ps lx output:

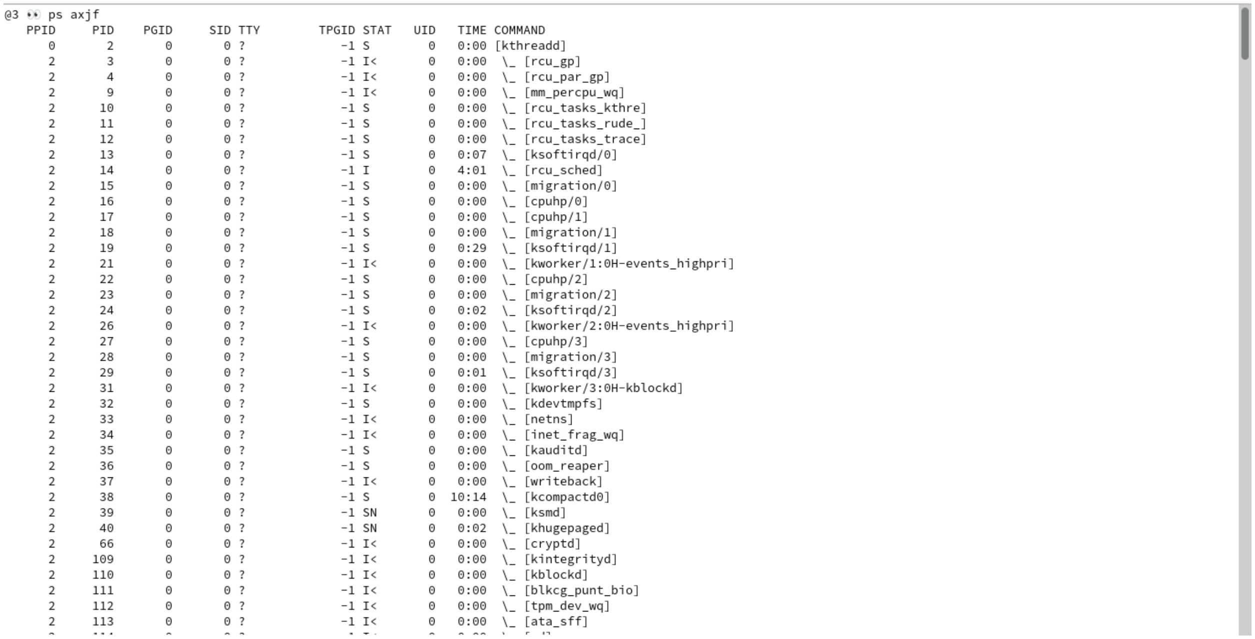


```
@3 👀 ps lx
F   UID     PID    PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4  1000    1700       1  20   0  23452 12396 -      Ss   ?          0:14 /usr/lib/systemd/systemd --user
5  1000    1706    1700  20   0 200236  1556 -      S    ?          0:00 (sd-pam)
1  1000    1728       1  20   0 526564  6272 -      SLl  ?          0:09 /usr/bin/gnome-keyring-daemon --daemonize --login
4  1000    1732    1684  20   0 374164  5028 -      Ssl+ tty2       0:00 /usr/libexec/gdm-wayland-session /usr/bin/gnome-session
0  1000    1735    1700  20   0  18844  3816 -      Ss   ?          0:00 /usr/bin/dbus-broker-launch --scope user
0  1000    1742    1735  20   0   9648  6868 -      S    ?          0:50 dbus-broker --log 4 --controller 10 --machine-id 370b754566864e688eca6025097dbfac --max-bytes 1
0  1000    1745    1732  20   0 476520  8176 -      Sl+  tty2       0:00 /usr/libexec/gnome-session-binary
0  1000    1809    1700  20   0 303836  4092 -      Ssl  ?          0:00 /usr/libexec/gnome-session-ctl --monitor
0  1000    1810    1700  20   0 159412  6316 -      Ssl  ?          0:00 /usr/libexec/uresourced --user
0  1000    1814    1700  20   0 781464 10680 -      Ssl  ?          0:12 /usr/libexec/gnome-session-binary --systemd-service --session=gnome
4  1000    1839    1700  20   0 5360572 328676 -    Ssl  ?        349:05 /usr/bin/gnome-shell
0  1000    1903    1700  20   0 308748  6988 -      Ssl  ?          0:00 /usr/libexec/at-spi-bus-launcher
0  1000    1910    1903  20   0   9624  3492 -      S    ?          0:00 /usr/bin/dbus-broker-launch --config-file=/usr/share/defaults/at-spi2/accessibility.conf --scop
0  1000    1912    1910  20   0   5288  2832 -      S    ?          0:01 dbus-broker --log 4 --controller 9 --machine-id 370b754566864e688eca6025097dbfac --max-bytes 10
0  1000    1928    1700  20   0 452280  6708 -      Ssl  ?          0:00 /usr/libexec/gvfsd
0  1000    1929    1700  20   0 448596  6028 -      Ssl  ?          0:00 /usr/libexec/xdg-permission-store
0  1000    1937    1700  20   0 379936  5380 -      Sl   ?          0:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f
0  1000    1960    1700  20   0 863000 13276 -      Ssl  ?          0:03 /usr/libexec/gnome-shell-calendar-server
0  1000    1961    1700  20   0 366384 23560 -      Ssl  ?         51:40 /usr/bin/pipewire
4  1000    1962    1700  20   0 333428 90520 -      SLsl ?         78:25 /usr/bin/pipewire-pulse
0  1000    1969    1700  20   0 156888  6380 -      Ssl  ?          0:02 /usr/libexec/dconf-service
0  1000    1970    1961  20   0 253104  7256 -      Sl   ?          0:05 /usr/bin/pipewire-media-session
0  1000    1983    1700  20   0 543028  9096 -      Ssl  ?          0:13 /usr/libexec/gvfs-udisks2-volume-monitor
0  1000    1998    1700  20   0 448364  6104 -      Ssl  ?          0:00 /usr/libexec/gvfs-mtp-volume-monitor
0  1000    2002    1700  20   0 450616  5836 -      Ssl  ?          0:00 /usr/libexec/gvfs-gphoto2-volume-monitor
0  1000    2006    1700  20   0 526884  7136 -      Ssl  ?          0:08 /usr/libexec/gvfs-afc-volume-monitor
0  1000    2011    1700  20   0 449028  6308 -      Ssl  ?          0:00 /usr/libexec/gvfs-goa-volume-monitor
0  1000    2014    1700  20   0 1115440 73632 -     SLsl ?          1:27 /usr/libexec/goa-daemon
0  1000    2020    1700  20   0 1235312 27812 -     Ssl  ?          0:07 /usr/libexec/evolution-source-registry
0  1000    2025    1700  20   0 376728  7208 -      Ssl  ?          0:22 /usr/libexec/gvfsd-metadata
0  1000    2035    1700  20   0 2174192 60728 -     Ssl  ?          0:40 /usr/libexec/evolution-calendar-factory
0  1000    2048    1700  20   0 2054348 31788 -     Ssl  ?          0:09 /usr/libexec/evolution-addressbook-factory
0  1000    2078    1700  20   0 455824  6848 -      Ssl  ?          0:35 /usr/libexec/goa-identity-service
0  1000    2092    1700  20   0 161756  6624 -      Ssl  ?          0:02 /usr/libexec/at-spi2-registryd --use-gnome-session
```

ps tree output:

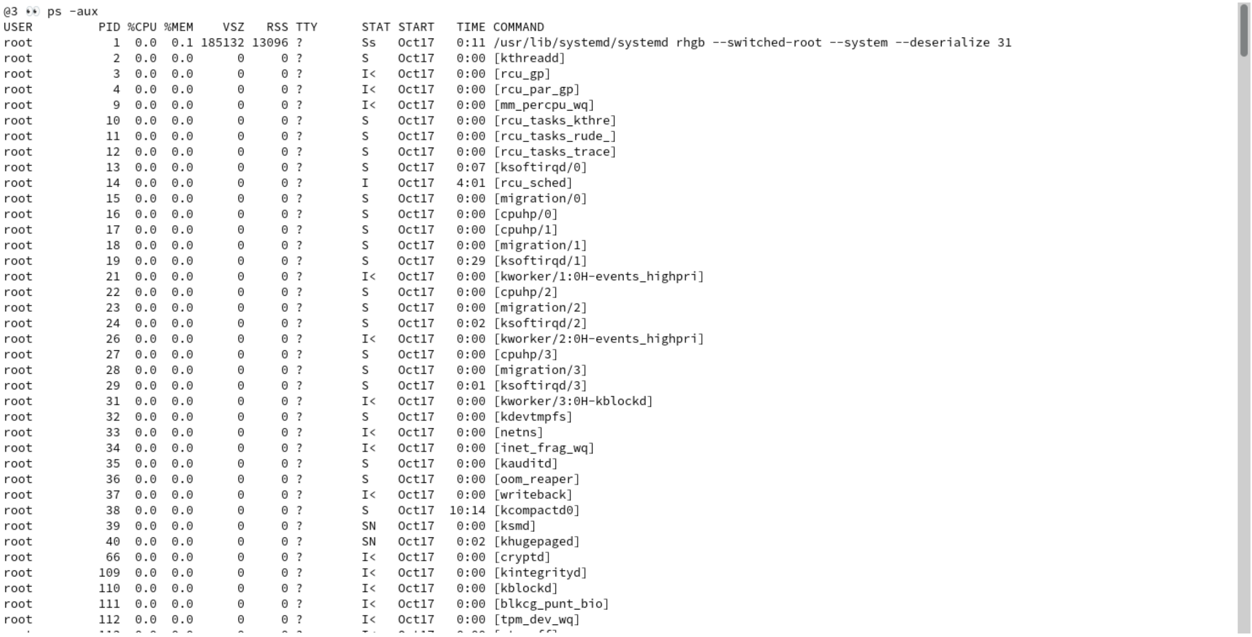
This command was not present in my system(fedora 34). When I looked into man pages of ps, I found out there are different "styles" of passing the options. Currently, this command accepts the unix version of options(which may be grouped and must be preceded by a dash), BSD options(which may be grouped and must not be used with a dash) and GNU long options(which are preceded by two dashes).

In my system, the command to print the process tree is ps -axjf. Its output is as follows.



```
@3 👀 ps axjf
   PPID     PID    PGID     SID TTY        TPGID STAT   UID   TIME COMMAND
      0       2       0       0 ?             -1 S        0   0:00 [kthreadd]
      2       3       0       0 ?             -1 I<       0   0:00  \_ [rcu_gp]
      2       4       0       0 ?             -1 I<       0   0:00  \_ [rcu_par_gp]
      2       9       0       0 ?             -1 I<       0   0:00  \_ [mm_percpu_wq]
      2      10       0       0 ?             -1 S        0   0:00  \_ [rcu_tasks_kthre]
      2      11       0       0 ?             -1 S        0   0:00  \_ [rcu_tasks_rude_]
      2      12       0       0 ?             -1 S        0   0:00  \_ [rcu_tasks_trace]
      2      13       0       0 ?             -1 S        0   0:07  \_ [ksoftirqd/0]
      2      14       0       0 ?             -1 I        0   4:01  \_ [rcu_sched]
      2      15       0       0 ?             -1 S        0   0:00  \_ [migration/0]
      2      16       0       0 ?             -1 S        0   0:00  \_ [cpuhp/0]
      2      17       0       0 ?             -1 S        0   0:00  \_ [cpuhp/1]
      2      18       0       0 ?             -1 S        0   0:00  \_ [migration/1]
      2      19       0       0 ?             -1 S        0   0:29  \_ [ksoftirqd/1]
      2      21       0       0 ?             -1 I<       0   0:00  \_ [kworker/1:0H-events_highpri]
      2      22       0       0 ?             -1 S        0   0:00  \_ [cpuhp/2]
      2      23       0       0 ?             -1 S        0   0:00  \_ [migration/2]
      2      24       0       0 ?             -1 S        0   0:02  \_ [ksoftirqd/2]
      2      26       0       0 ?             -1 I<       0   0:00  \_ [kworker/2:0H-events_highpri]
      2      27       0       0 ?             -1 S        0   0:00  \_ [cpuhp/3]
      2      28       0       0 ?             -1 S        0   0:00  \_ [migration/3]
      2      29       0       0 ?             -1 S        0   0:01  \_ [ksoftirqd/3]
      2      31       0       0 ?             -1 I<       0   0:00  \_ [kworker/3:0H-kblockd]
      2      32       0       0 ?             -1 S        0   0:00  \_ [kdevtmpfs]
      2      33       0       0 ?             -1 I<       0   0:00  \_ [netns]
      2      34       0       0 ?             -1 I<       0   0:00  \_ [inet_frag_wq]
      2      35       0       0 ?             -1 S        0   0:00  \_ [kauditd]
      2      36       0       0 ?             -1 S        0   0:00  \_ [oom_reaper]
      2      37       0       0 ?             -1 I<       0   0:00  \_ [writeback]
      2      38       0       0 ?             -1 S        0  10:14  \_ [kcompactd0]
      2      39       0       0 ?             -1 SN       0   0:00  \_ [ksmd]
      2      40       0       0 ?             -1 SN       0   0:02  \_ [khugepaged]
      2      66       0       0 ?             -1 I<       0   0:00  \_ [cryptd]
      2     109       0       0 ?             -1 I<       0   0:00  \_ [kintegrityd]
      2     110       0       0 ?             -1 I<       0   0:00  \_ [kblockd]
      2     111       0       0 ?             -1 I<       0   0:00  \_ [blkcg_punt_bio]
      2     112       0       0 ?             -1 I<       0   0:00  \_ [tpm_dev_wq]
      2     113       0       0 ?             -1 I<       0   0:00  \_ [ata_sff]
```

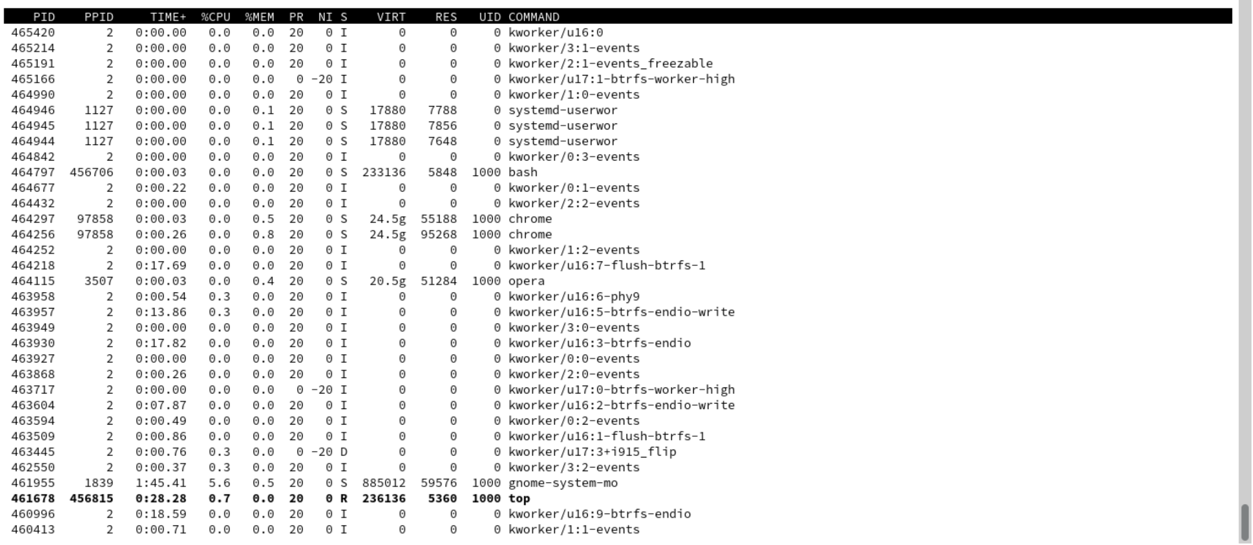
ps -aux output:



```
@3 👀 ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 185132 13096 ?        Ss   Oct17   0:11 /usr/lib/systemd/systemd rhgb --switched-root --system --deserialize 31
root           2  0.0  0.0      0     0 ?        S    Oct17   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Oct17   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Oct17   0:00 [rcu_par_gp]
root           9  0.0  0.0      0     0 ?        I<   Oct17   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    Oct17   0:00 [rcu_tasks_kthre]
root          11  0.0  0.0      0     0 ?        S    Oct17   0:00 [rcu_tasks_rude_]
root          12  0.0  0.0      0     0 ?        S    Oct17   0:00 [rcu_tasks_trace]
root          13  0.0  0.0      0     0 ?        S    Oct17   0:07 [ksoftirqd/0]
root          14  0.0  0.0      0     0 ?        I    Oct17   4:01 [rcu_sched]
root          15  0.0  0.0      0     0 ?        S    Oct17   0:00 [migration/0]
root          16  0.0  0.0      0     0 ?        S    Oct17   0:00 [cpuhp/0]
root          17  0.0  0.0      0     0 ?        S    Oct17   0:00 [cpuhp/1]
root          18  0.0  0.0      0     0 ?        S    Oct17   0:00 [migration/1]
root          19  0.0  0.0      0     0 ?        S    Oct17   0:29 [ksoftirqd/1]
root          21  0.0  0.0      0     0 ?        I<   Oct17   0:00 [kworker/1:0H-events_highpri]
root          22  0.0  0.0      0     0 ?        S    Oct17   0:00 [cpuhp/2]
root          23  0.0  0.0      0     0 ?        S    Oct17   0:00 [migration/2]
root          24  0.0  0.0      0     0 ?        S    Oct17   0:02 [ksoftirqd/2]
root          26  0.0  0.0      0     0 ?        I<   Oct17   0:00 [kworker/2:0H-events_highpri]
root          27  0.0  0.0      0     0 ?        S    Oct17   0:00 [cpuhp/3]
root          28  0.0  0.0      0     0 ?        S    Oct17   0:00 [migration/3]
root          29  0.0  0.0      0     0 ?        S    Oct17   0:01 [ksoftirqd/3]
root          31  0.0  0.0      0     0 ?        I<   Oct17   0:00 [kworker/3:0H-kblockd]
root          32  0.0  0.0      0     0 ?        S    Oct17   0:00 [kdevtmpfs]
root          33  0.0  0.0      0     0 ?        I<   Oct17   0:00 [netns]
root          34  0.0  0.0      0     0 ?        I<   Oct17   0:00 [inet_frag_wq]
root          35  0.0  0.0      0     0 ?        S    Oct17   0:00 [kauditd]
root          36  0.0  0.0      0     0 ?        S    Oct17   0:00 [oom_reaper]
root          37  0.0  0.0      0     0 ?        I<   Oct17   0:00 [writeback]
root          38  0.0  0.0      0     0 ?        S    Oct17  10:14 [kcompactd0]
root          39  0.0  0.0      0     0 ?        SN   Oct17   0:00 [ksmd]
root          40  0.0  0.0      0     0 ?        SN   Oct17   0:02 [khugepaged]
root          66  0.0  0.0      0     0 ?        I<   Oct17   0:00 [cryptd]
root         109  0.0  0.0      0     0 ?        I<   Oct17   0:00 [kintegrityd]
root         110  0.0  0.0      0     0 ?        I<   Oct17   0:00 [kblockd]
root         111  0.0  0.0      0     0 ?        I<   Oct17   0:00 [blkcg_punt_bio]
root         112  0.0  0.0      0     0 ?        I<   Oct17   0:00 [tpm_dev_wq]
```

2. Learn the top command to display the resource utilization statistics of processes:
• Open a terminal and type the top command
• Start a browser and see the effect on the top display
• Compile a C program and observe the same effect (Use a long loop - say while(1) to observe the effect)
• From the top display, answer the following:
– How much memory is free in the system?
– Which process is taking more CPU?
– Which process has got maximum memory share?
• Write a CPU bound C program and a I/O bound C program (e.g. using more printf statements within while(1) loop), compile and execute both of them.

1. Open a terminal and type the top command



```
    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
 465420       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/u16:0
 465214       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/3:1-events
 465191       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/2:1-events_freezable
 465166       2   0:00.00   0.0   0.0   0 -20 I       0      0     0 kworker/u17:1-btrfs-worker-high
 464990       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/1:0-events
 464946    1127   0:00.00   0.0   0.1  20   0 S   17880   7788     0 systemd-userwor
 464945    1127   0:00.00   0.0   0.1  20   0 S   17880   7856     0 systemd-userwor
 464944    1127   0:00.00   0.0   0.1  20   0 S   17880   7648     0 systemd-userwor
 464842       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/0:3-events
 464797  456706   0:00.03   0.0   0.0  20   0 S  233136   5848  1000 bash
 464677       2   0:00.22   0.0   0.0  20   0 I       0      0     0 kworker/0:1-events
 464432       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/2:2-events
 464297   97858   0:00.03   0.0   0.5  20   0 S   24.5g  55188  1000 chrome
 464256   97858   0:00.26   0.0   0.8  20   0 S   24.5g  95268  1000 chrome
 464252       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/1:2-events
 464218       2   0:17.69   0.0   0.0  20   0 I       0      0     0 kworker/u16:7-flush-btrfs-1
 464115    3507   0:00.03   0.0   0.4  20   0 S   20.5g  51284  1000 opera
 463958       2   0:00.54   0.3   0.0  20   0 I       0      0     0 kworker/u16:6-phy9
 463957       2   0:13.86   0.3   0.0  20   0 I       0      0     0 kworker/u16:5-btrfs-endio-write
 463949       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/3:0-events
 463930       2   0:17.82   0.0   0.0  20   0 I       0      0     0 kworker/u16:3-btrfs-endio
 463927       2   0:00.00   0.0   0.0  20   0 I       0      0     0 kworker/0:0-events
 463868       2   0:00.26   0.0   0.0  20   0 I       0      0     0 kworker/2:0-events
 463717       2   0:00.00   0.0   0.0   0 -20 I       0      0     0 kworker/u17:0-btrfs-worker-high
 463604       2   0:07.87   0.0   0.0  20   0 I       0      0     0 kworker/u16:2-btrfs-endio-write
 463594       2   0:00.49   0.0   0.0  20   0 I       0      0     0 kworker/0:2-events
 463509       2   0:00.86   0.0   0.0  20   0 I       0      0     0 kworker/u16:1-flush-btrfs-1
 463445       2   0:00.76   0.3   0.0   0 -20 D       0      0     0 kworker/u17:3+i915_flip
 462550       2   0:00.37   0.3   0.0  20   0 I       0      0     0 kworker/3:2-events
 461955    1839   1:45.41   5.6   0.5  20   0 S  885012  59576  1000 gnome-system-mo
 461678  456815   0:28.28   0.7   0.0  20   0 R  236136   5360  1000 top
 460996       2   0:18.59   0.0   0.0  20   0 I       0      0     0 kworker/u16:9-btrfs-endio
 460413       2   0:00.71   0.0   0.0  20   0 I       0      0     0 kworker/1:1-events
```

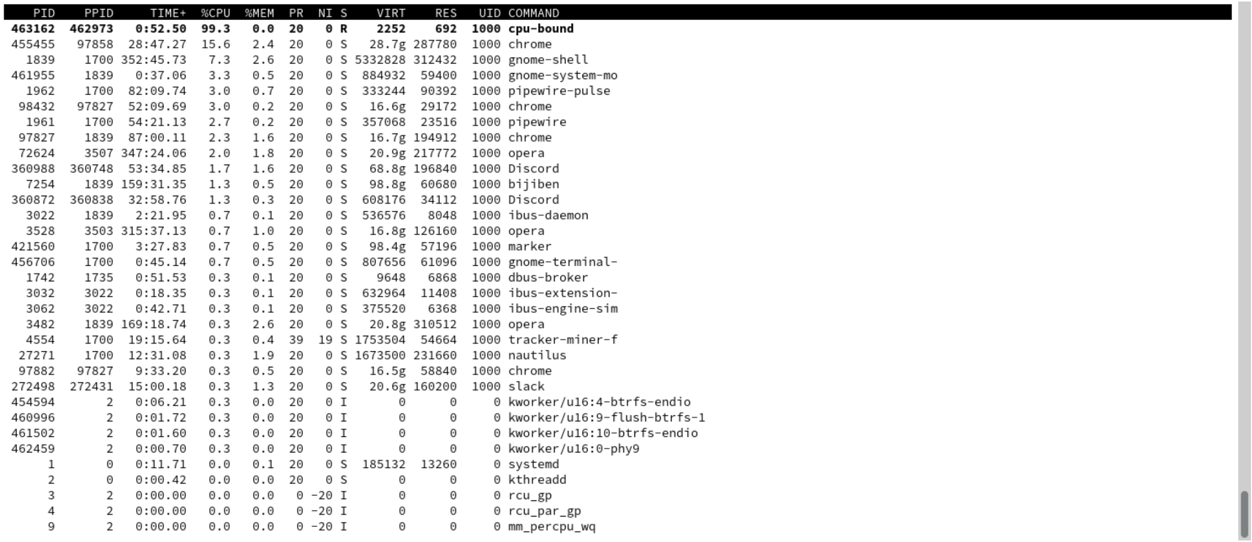
2. Compile a C program and observe the same effect (Use a long loop - say while(1) to observe the effect)

We can see that this program is contributing to about 99% usage of CPU. This is not surprising as the program contains an infinite loop which is taking CPU time and hence the CPU usage share is so high for this program.

The code is as follows.

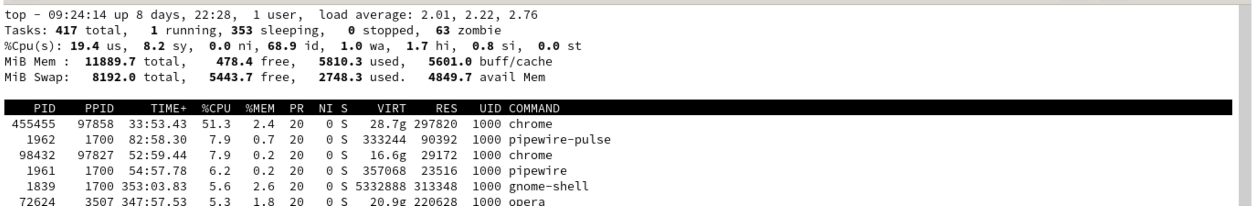
cpu-bound.c file

```
int main() {
        while (1) {
        }
        return 0;
}
```



```
    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
 463162  462973   0:52.50  99.3   0.0  20   0 R    2252    692  1000 cpu-bound
 455455   97858  28:47.27  15.6   2.4  20   0 S   28.7g 287780  1000 chrome
   1839    1700 352:45.73   7.3   2.6  20   0 S 5332828 312432  1000 gnome-shell
 461955    1839   0:37.06   3.3   0.5  20   0 S  884932  59400  1000 gnome-system-mo
   1962    1700  82:09.74   3.0   0.7  20   0 S  333244  90392  1000 pipewire-pulse
  98432   97827  52:09.69   3.0   0.2  20   0 S   16.6g  29172  1000 chrome
   1961    1700  54:21.13   2.7   0.2  20   0 S  357068  23516  1000 pipewire
  97827    1839  87:00.11   2.3   1.6  20   0 S   16.7g 194912  1000 chrome
  72624    3507 347:24.06   2.0   1.8  20   0 S   20.9g 217772  1000 opera
 360988  360748  53:34.85   1.7   1.6  20   0 S   68.8g 196840  1000 Discord
   7254    1839 159:31.35   1.3   0.5  20   0 S   98.8g  60680  1000 bijiben
 360872  360838  32:58.76   1.3   0.3  20   0 S  608176  34112  1000 Discord
   3022    1839   2:21.95   0.7   0.1  20   0 S  536576   8048  1000 ibus-daemon
   3528    3503 315:37.13   0.7   1.0  20   0 S   16.8g 126160  1000 opera
 421560    1700   3:27.83   0.7   0.5  20   0 S   98.4g  57196  1000 marker
 456706    1700   0:45.14   0.7   0.5  20   0 S  807656  61096  1000 gnome-terminal-
   1742    1735   0:51.53   0.3   0.1  20   0 S    9648   6868  1000 dbus-broker
   3032    3022   0:18.35   0.3   0.1  20   0 S  632964  11408  1000 ibus-extension-
   3062    3022   0:42.71   0.3   0.1  20   0 S  375520   6368  1000 ibus-engine-sim
   3482    1839 169:18.74   0.3   2.6  20   0 S   20.8g 310512  1000 opera
   4554    1700  19:15.64   0.3   0.4  39  19 S 1753504  54664  1000 tracker-miner-f
  27271    1700  12:31.08   0.3   1.9  20   0 S 1673500 231660  1000 nautilus
  97882   97827   9:33.20   0.3   0.5  20   0 S   16.5g  58840  1000 chrome
 272498  272431  15:00.18   0.3   1.3  20   0 S   20.6g 160200  1000 slack
 454594       2   0:06.21   0.3   0.0  20   0 I       0      0     0 kworker/u16:4-btrfs-endio
 460996       2   0:01.72   0.3   0.0  20   0 I       0      0     0 kworker/u16:9-flush-btrfs-1
 461502       2   0:01.60   0.3   0.0  20   0 I       0      0     0 kworker/u16:10-btrfs-endio
 462459       2   0:00.70   0.3   0.0  20   0 I       0      0     0 kworker/u16:0-phy9
      1       0   0:11.71   0.0   0.1  20   0 S  185132  13260     0 systemd
      2       0   0:00.42   0.0   0.0  20   0 S       0      0     0 kthreadd
      3       2   0:00.00   0.0   0.0   0 -20 I       0      0     0 rcu_gp
      4       2   0:00.00   0.0   0.0   0 -20 I       0      0     0 rcu_par_gp
      9       2   0:00.00   0.0   0.0   0 -20 I       0      0     0 mm_percpu_wq
```

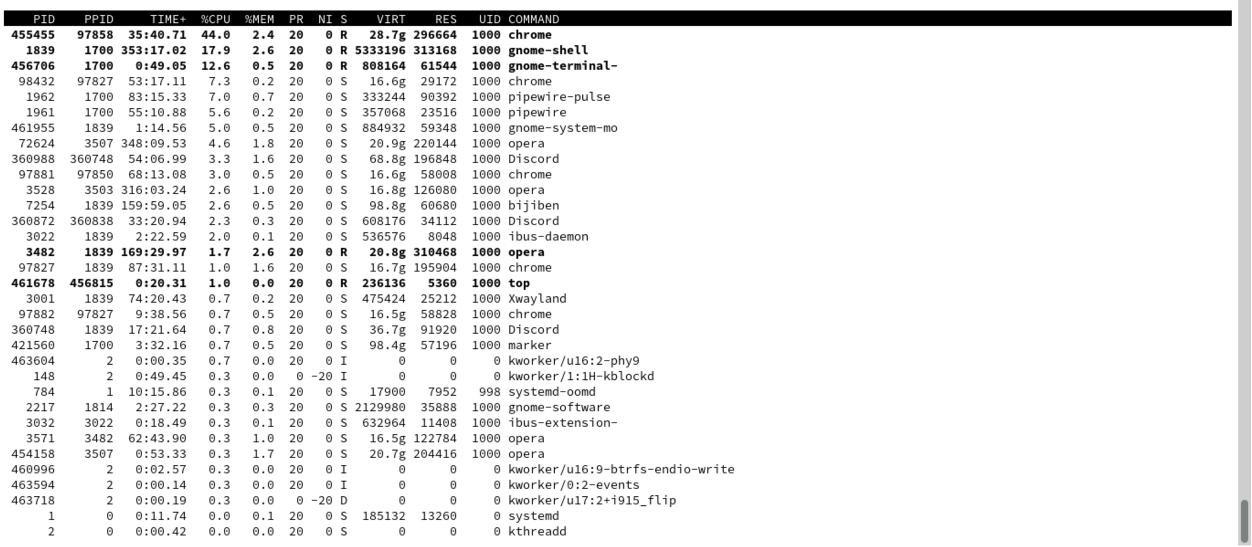
3. How much memory is free in the system?



```
top - 09:24:14 up 8 days, 22:28,  1 user,  load average: 2.01, 2.22, 2.76
Tasks: 417 total,   1 running, 353 sleeping,   0 stopped,  63 zombie
%Cpu(s): 19.4 us,  8.2 sy,  0.0 ni, 68.9 id,  1.0 wa,  1.7 hi,  0.8 si,  0.0 st
MiB Mem :  11889.7 total,    478.4 free,   5810.3 used,   5601.0 buff/cache
MiB Swap:   8192.0 total,   5443.7 free,   2748.3 used.   4849.7 avail Mem

    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
 455455   97858  33:53.43  51.3   2.4  20   0 S   28.7g 297820  1000 chrome
   1962    1700  82:58.30   7.9   0.7  20   0 S  333244  90392  1000 pipewire-pulse
  98432   97827  52:59.44   7.9   0.2  20   0 S   16.6g  29172  1000 chrome
   1961    1700  54:57.78   6.2   0.2  20   0 S  357068  23516  1000 pipewire
   1839    1700 353:03.83   5.6   2.6  20   0 S 5332888 313348  1000 gnome-shell
  72624    3507 347:57.53   5.3   1.8  20   0 S   20.9g 220628  1000 opera
```

From the above image, we can see that about 458 MB memory is free currently.

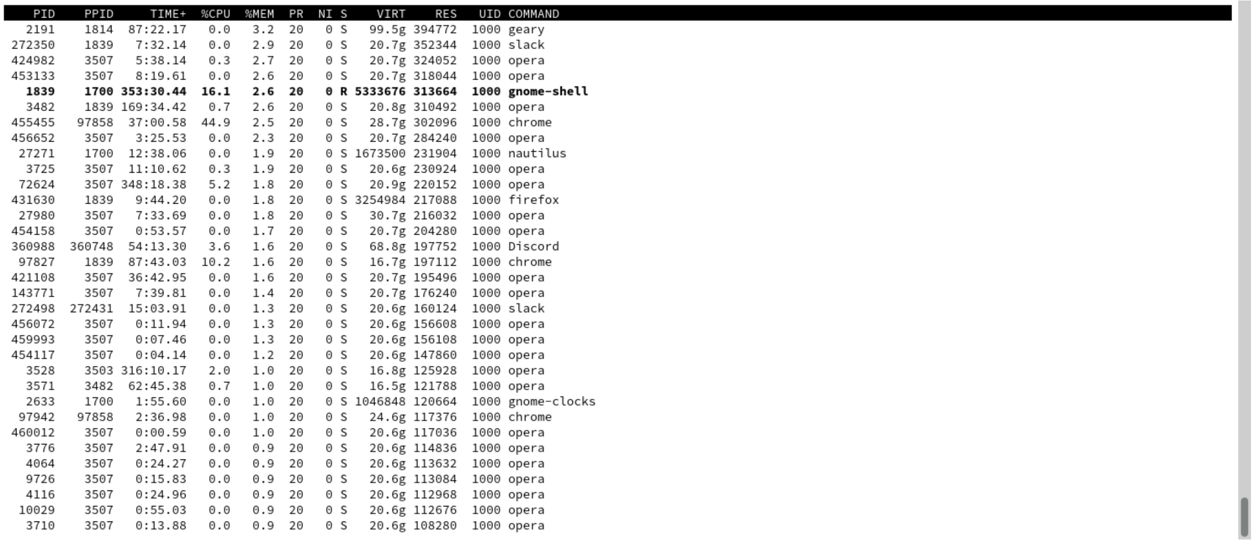
4. Which process is taking more CPU?



```
    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
 455455   97858  35:40.71  44.0   2.4  20   0 R   28.7g 296664  1000 chrome
   1839    1700 353:17.02  17.9   2.6  20   0 R 5333196 313168  1000 gnome-shell
 456706    1700   0:49.05  12.6   0.5  20   0 R  808164  61544  1000 gnome-terminal-
  98432   97827  53:17.11   7.3   0.2  20   0 S   16.6g  29172  1000 chrome
   1962    1700  83:15.33   7.0   0.7  20   0 S  333244  90392  1000 pipewire-pulse
   1961    1700  55:10.88   5.6   0.2  20   0 S  357068  23516  1000 pipewire
 461955    1839   1:14.56   5.0   0.5  20   0 S  884932  59348  1000 gnome-system-mo
  72624    3507 348:09.53   4.6   1.8  20   0 S   20.9g 220144  1000 opera
 360988  360748  54:06.99   3.3   1.6  20   0 S   68.8g 196848  1000 Discord
  97881   97850  68:13.08   3.0   0.5  20   0 S   16.6g  58008  1000 chrome
   3528    3503 316:03.24   2.6   1.0  20   0 S   16.8g 126080  1000 opera
   7254    1839 159:59.05   2.6   0.5  20   0 S   98.8g  60680  1000 bijiben
 360872  360838  33:20.94   2.3   0.3  20   0 S  608176  34112  1000 Discord
   3022    1839   2:22.59   2.0   0.1  20   0 S  536576   8048  1000 ibus-daemon
   3482    1839 169:29.97   1.7   2.6  20   0 R   20.8g 310468  1000 opera
  97827    1839  87:31.11   1.0   1.6  20   0 S   16.7g 195904  1000 chrome
 461678  456815   0:20.31   1.0   0.0  20   0 R  236136   5360  1000 top
   3001    1839  74:20.43   0.7   0.2  20   0 S  475424  25212  1000 Xwayland
  97882   97827   9:38.56   0.7   0.5  20   0 S   16.5g  58828  1000 chrome
 360748    1839  17:21.64   0.7   0.8  20   0 S   36.7g  91920  1000 Discord
 421560    1700   3:32.16   0.7   0.5  20   0 S   98.4g  57196  1000 marker
 463604       2   0:00.35   0.7   0.0  20   0 I       0      0     0 kworker/u16:2-phy9
    148       2   0:49.45   0.3   0.0   0 -20 I       0      0     0 kworker/1:1H-kblockd
    784       1  10:15.86   0.3   0.1  20   0 S   17900   7952   998 systemd-oomd
   2217    1814   2:27.22   0.3   0.3  20   0 S 2129980  35888  1000 gnome-software
   3032    3022   0:18.49   0.3   0.1  20   0 S  632964  11408  1000 ibus-extension-
   3571    3482  62:43.90   0.3   1.0  20   0 S   16.5g 122784  1000 opera
 454158    3507   0:53.33   0.3   1.7  20   0 S   20.7g 204416  1000 opera
 460996       2   0:02.57   0.3   0.0  20   0 I       0      0     0 kworker/u16:9-btrfs-endio-write
 463594       2   0:00.14   0.3   0.0  20   0 I       0      0     0 kworker/0:2-events
 463718       2   0:00.19   0.3   0.0   0 -20 D       0      0     0 kworker/u17:2+i915_flip
      1       0   0:11.74   0.0   0.1  20   0 S  185132  13260     0 systemd
      2       0   0:00.42   0.0   0.0  20   0 S       0      0     0 kthreadd
```

Google chrome is taking top share of CPU usage.

5. Which process has got maximum memory share ?



```
    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
   2191    1814  87:22.17   0.0   3.2  20   0 S   99.5g 394772  1000 geary
 272350    1839   7:32.14   0.0   2.9  20   0 S   20.7g 352344  1000 slack
 424982    3507   5:38.14   0.3   2.7  20   0 S   20.7g 324052  1000 opera
 453133    3507   8:19.61   0.0   2.6  20   0 S   20.7g 318044  1000 opera
   1839    1700 353:30.44  16.1   2.6  20   0 R 5333676 313664  1000 gnome-shell
   3482    1839 169:34.42   0.7   2.6  20   0 S   20.8g 310492  1000 opera
 455455   97858  37:00.58  44.9   2.5  20   0 S   28.7g 302096  1000 chrome
 456652    3507   3:25.53   0.0   2.3  20   0 S   20.7g 284240  1000 opera
  27271    1700  12:38.06   0.0   1.9  20   0 S 1673500 231904  1000 nautilus
   3725    3507  11:10.62   0.3   1.9  20   0 S   20.6g 230924  1000 opera
  72624    3507 348:18.38   5.2   1.8  20   0 S   20.9g 220152  1000 opera
 431630    1839   9:44.20   0.0   1.8  20   0 S 3254984 217088  1000 firefox
  27980    3507   7:33.69   0.0   1.8  20   0 S   30.7g 216032  1000 opera
 454158    3507   0:53.57   0.0   1.7  20   0 S   20.7g 204280  1000 opera
 360988  360748  54:13.30   3.6   1.6  20   0 S   68.8g 197752  1000 Discord
  97827    1839  87:43.03  10.2   1.6  20   0 S   16.7g 197112  1000 chrome
 421108    3507  36:42.95   0.0   1.6  20   0 S   20.7g 195496  1000 opera
 143771    3507   7:39.81   0.0   1.4  20   0 S   20.7g 176240  1000 opera
 272498  272431  15:03.91   0.0   1.3  20   0 S   20.6g 160124  1000 slack
 456072    3507   0:11.94   0.0   1.3  20   0 S   20.6g 156608  1000 opera
 459993    3507   0:07.46   0.0   1.3  20   0 S   20.6g 156108  1000 opera
 454117    3507   0:04.14   0.0   1.2  20   0 S   20.6g 147860  1000 opera
   3528    3503 316:10.17   2.0   1.0  20   0 S   16.8g 125928  1000 opera
   3571    3482  62:45.38   0.7   1.0  20   0 S   16.5g 121788  1000 opera
   2633    1700   1:55.60   0.0   1.0  20   0 S 1046848 120664  1000 gnome-clocks
  97942   97858   2:36.98   0.0   1.0  20   0 S   24.6g 117376  1000 chrome
 460012    3507   0:00.59   0.0   1.0  20   0 S   20.6g 117036  1000 opera
   3776    3507   2:47.91   0.0   0.9  20   0 S   20.6g 114836  1000 opera
   4064    3507   0:24.27   0.0   0.9  20   0 S   20.6g 113632  1000 opera
   9726    3507   0:15.83   0.0   0.9  20   0 S   20.6g 113084  1000 opera
   4116    3507   0:24.96   0.0   0.9  20   0 S   20.6g 112968  1000 opera
  10029    3507   0:55.03   0.0   0.9  20   0 S   20.6g 112676  1000 opera
   3710    3507   0:13.88   0.0   0.9  20   0 S   20.6g 108280  1000 opera
```

Geary has got maximum memory share of 3.2%

6. Write a CPU bound C program and a I/O bound C program (e.g. using more printf statements within while(1) loop), compile and execute both of them.

The cpu bound and io bound program are as follows.

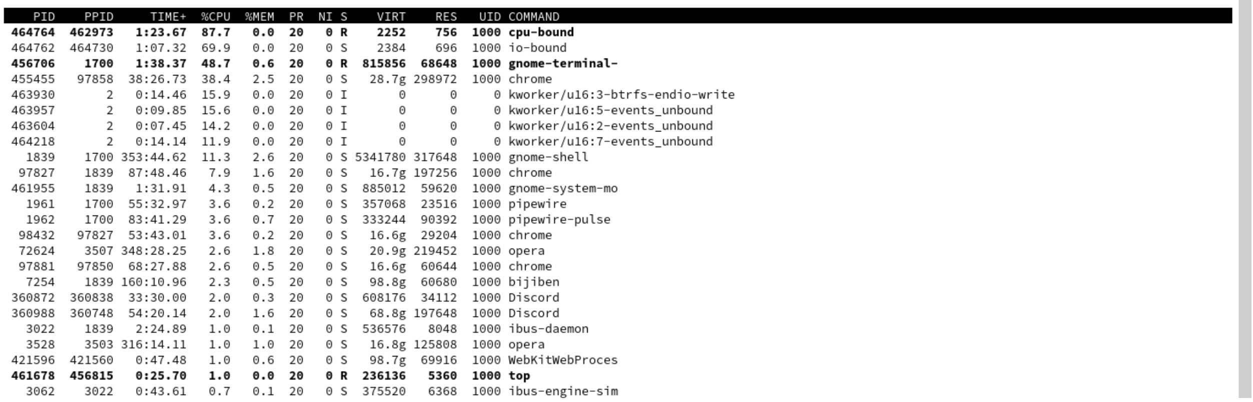
cpu-bound.c file:

```
int main() {
        while (1) {

        }
        return 0;
}
```

io-bound.c file:

```
#include<stdio.h>

int main() {
        while (1) {
                printf("Hello\n");
        }
        return 0;
}
```

We can see that both the process takes a lot of share of CPU resources.

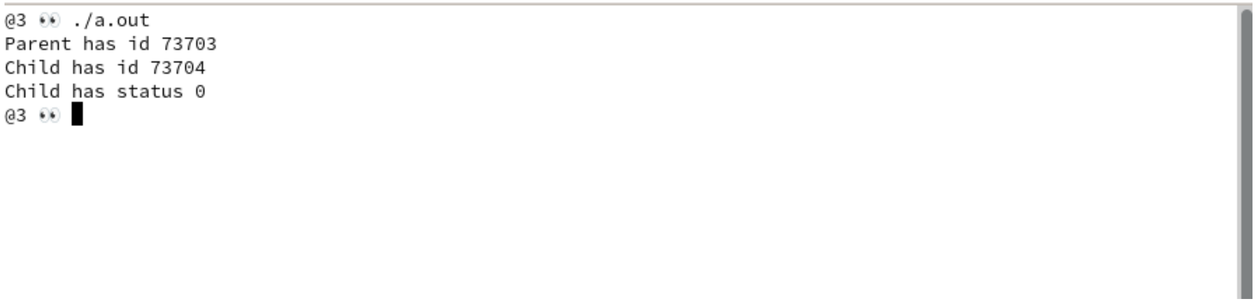


```
    PID    PPID     TIME+  %CPU  %MEM  PR  NI S    VIRT    RES   UID COMMAND
 464764  462973   1:23.67  87.7   0.0  20   0 R    2252    756  1000 cpu-bound
 464762  464730   1:07.32  69.9   0.0  20   0 S    2384    696  1000 io-bound
 456706    1700   1:38.37  48.7   0.6  20   0 R  815856  68648  1000 gnome-terminal-
 455455   97858  38:26.73  38.4   2.5  20   0 S   28.7g 298972  1000 chrome
 463930       2   0:14.46  15.9   0.0  20   0 I       0      0     0 kworker/u16:3-btrfs-endio-write
 463957       2   0:09.85  15.6   0.0  20   0 I       0      0     0 kworker/u16:5-events_unbound
 463604       2   0:07.45  14.2   0.0  20   0 I       0      0     0 kworker/u16:2-events_unbound
 464218       2   0:14.14  11.9   0.0  20   0 I       0      0     0 kworker/u16:7-events_unbound
   1839    1700 353:44.62  11.3   2.6  20   0 S 5341780 317648  1000 gnome-shell
  97827    1839  87:48.46   7.9   1.6  20   0 S   16.7g 197256  1000 chrome
 461955    1839   1:31.91   4.3   0.5  20   0 S  885012  59620  1000 gnome-system-mo
   1961    1700  55:32.97   3.6   0.2  20   0 S  357068  23516  1000 pipewire
   1962    1700  83:41.29   3.6   0.7  20   0 S  333244  90392  1000 pipewire-pulse
  98432   97827  53:43.01   3.6   0.2  20   0 S   16.6g  29204  1000 chrome
  72624    3507 348:28.25   2.6   1.8  20   0 S   20.9g 219452  1000 opera
  97881   97850  68:27.88   2.6   0.5  20   0 S   16.6g  60644  1000 chrome
   7254    1839 160:10.96   2.3   0.5  20   0 S   98.8g  60680  1000 bijiben
 360872  360838  33:30.00   2.0   0.3  20   0 S  608176  34112  1000 Discord
 360988  360748  54:20.14   2.0   1.6  20   0 S   68.8g 197648  1000 Discord
   3022    1839   2:24.89   1.0   0.1  20   0 S  536576   8048  1000 ibus-daemon
   3528    3503 316:14.11   1.0   1.0  20   0 S   16.8g 125808  1000 opera
 421596  421560   0:47.48   1.0   0.6  20   0 S   98.7g  69916  1000 WebKitWebProces
 461678  456815   0:25.70   1.0   0.0  20   0 R  236136   5360  1000 top
   3062    3022   0:43.61   0.7   0.1  20   0 S  375520   6368  1000 ibus-engine-sim
```

3. Write a program in C that creates a child process, waits for the termination of the child and lists its PID, together with the state in which the process was terminated (in decimal and hexadecimal).

q3.c file:

```
#include<unistd.h>
#include<stdio.h>
#include<sys/wait.h>
#include<stdlib.h>

int main() {
        pid_t childId;
        int childStatus;
        if (fork() == 0)
                exit(0);
        else
                childId = wait(&childStatus);

        printf("Parent has id %d\n", getpid());
        printf("Child has id %d\n", childId);
        printf("Child has status %d\n", childStatus);
        return 0;
}
```



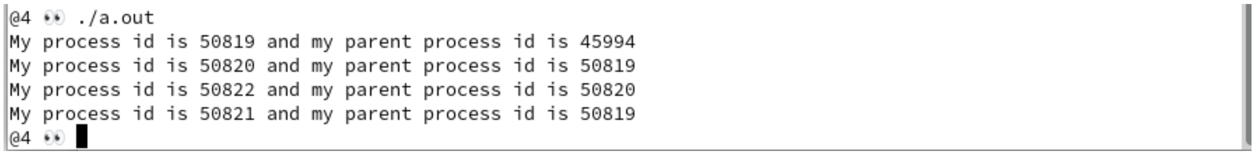
```
@3 👀 ./a.out
Parent has id 73703
Child has id 73704
Child has status 0
@3 👀 █
```

4. Write a C program such that it forks a new process. Then the parent process and the child process should create one more process such that the program in all has four running processes. Each process should print its process ID and its parent process ID. Draw process hierarchy starting from parent process.

q4.c file:

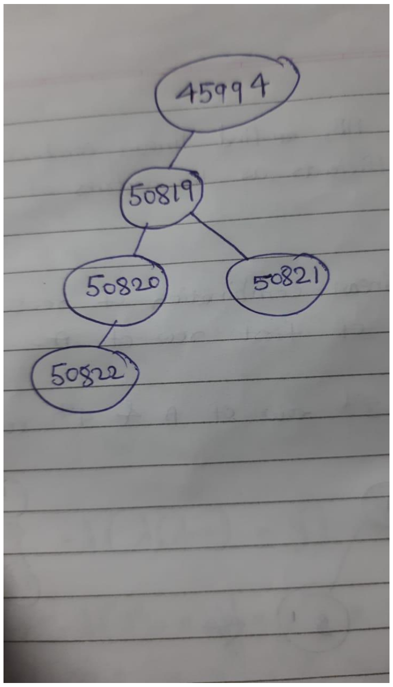
```
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>

int main() {
        fork();
        fork();
        printf("My process id is %d and my parent process id is %d\n", getpid(), getppid());
        return 0;
}
```

Output:



```
@4 👀 ./a.out
My process id is 50819 and my parent process id is 45994
My process id is 50820 and my parent process id is 50819
My process id is 50822 and my parent process id is 50820
My process id is 50821 and my parent process id is 50819
@4 👀 █
```

Process hierarchy:



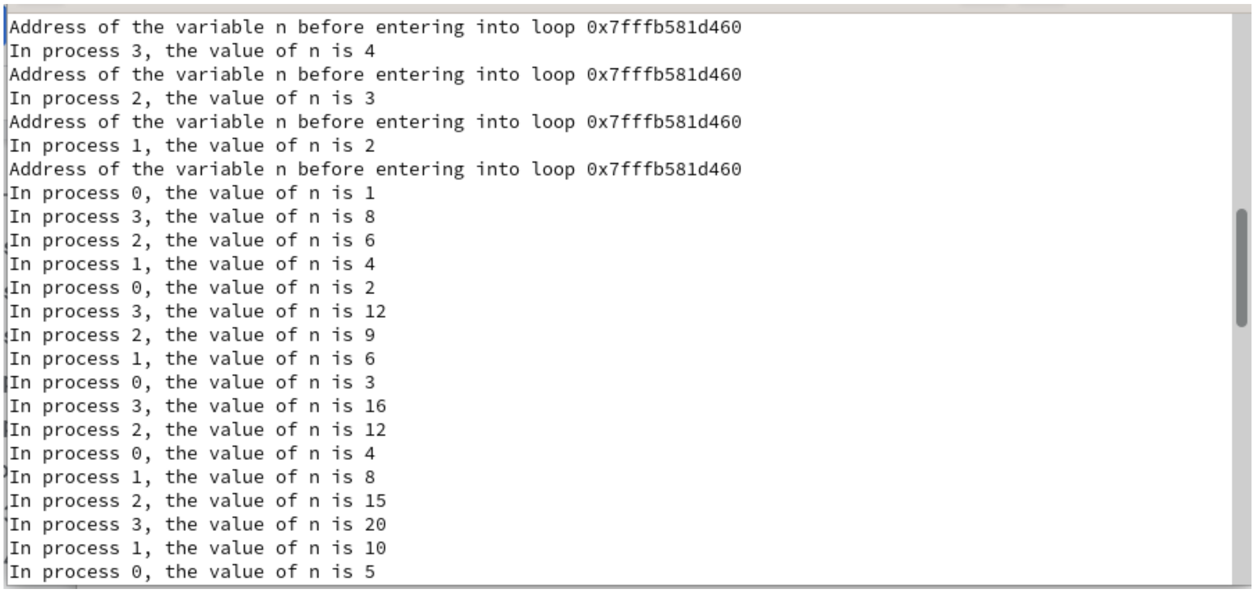
5. In a C program, print the address of the variable and enter into a long loop (say using while(1)).

- Start three to four processes of the same program and observe the printed address values.
- Show how two processes which are members of the relationship parent-child are concurrent from execution point of view, initially the child is copy of the parent, but every process has its own data.

q5.c file:

```
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>

int main() {
        int I1, I2;

        if (fork() == 0)
                I1 = 0;
        else
                I1 = 1;

        if (fork() == 0)
                I2 = 0;
        else
                I2 = 1;

        int n = 0;
        int processNo = 2 * I1 + I2;
        printf("Address of the variable n before entering into loop %p\n", &n);

        while (1) {
                if (I1 == 0 && I2 == 0) {
                        n += 1;
                } else if (I1 == 0 && I2 == 1) {
                        n += 2;
                } else if (I1 == 1 && I2 == 0) {
                        n += 3;
                } else {
                        n += 4;
                }
                printf("In process %d, the value of n is %d\n", processNo, n);
                sleep(2);
        }
        return 0;
}
```

Output:



```
Address of the variable n before entering into loop 0x7fffb581d460
In process 3, the value of n is 4
Address of the variable n before entering into loop 0x7fffb581d460
In process 2, the value of n is 3
Address of the variable n before entering into loop 0x7fffb581d460
In process 1, the value of n is 2
Address of the variable n before entering into loop 0x7fffb581d460
In process 0, the value of n is 1
In process 3, the value of n is 8
In process 2, the value of n is 6
In process 1, the value of n is 4
In process 0, the value of n is 2
In process 3, the value of n is 12
In process 2, the value of n is 9
In process 1, the value of n is 6
In process 0, the value of n is 3
In process 3, the value of n is 16
In process 2, the value of n is 12
In process 0, the value of n is 4
In process 1, the value of n is 8
In process 2, the value of n is 15
In process 3, the value of n is 20
In process 1, the value of n is 10
In process 0, the value of n is 5
```

We can see from the output that the value of variable n is different in each of the 4 processes but they have the same virtual address.

6. Test the source code below:

```
for(i = 1; i <= 10; i + +){
fork();
printf("The process with the PID=%d",getpid());
}
```

In the next phase, modify the code, such as after all created processes have finished execution, in a file process management.txt the total number of created processes should be stored.

q6.cpp file:

```
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>
#include<stdlib.h>
#include<fstream>
#include<iostream>

using namespace std;

int main() {
        for(int i = 1; i <= 10; i++){
                fork();
        }

        ofstream file;
        file.open("management.txt", ofstream::out | ofstream::trunc);
        file << 1024;
        file.close();
        return 0;
}
```

7. Write two programs: one called client.c, the other called server.c. The client program lists a prompter and reads from the keyboard two integers and one of the characters ‘+’ or ‘−’ . The read information is transmitted with the help of the system call excel to a child process, which executes the server code. After the child (server) process finishes the operation, it transmits the result to the parent process (client) with the help of the system call exit. The client process prints the result on the screen and also reprints the prompter, ready for a new reading.

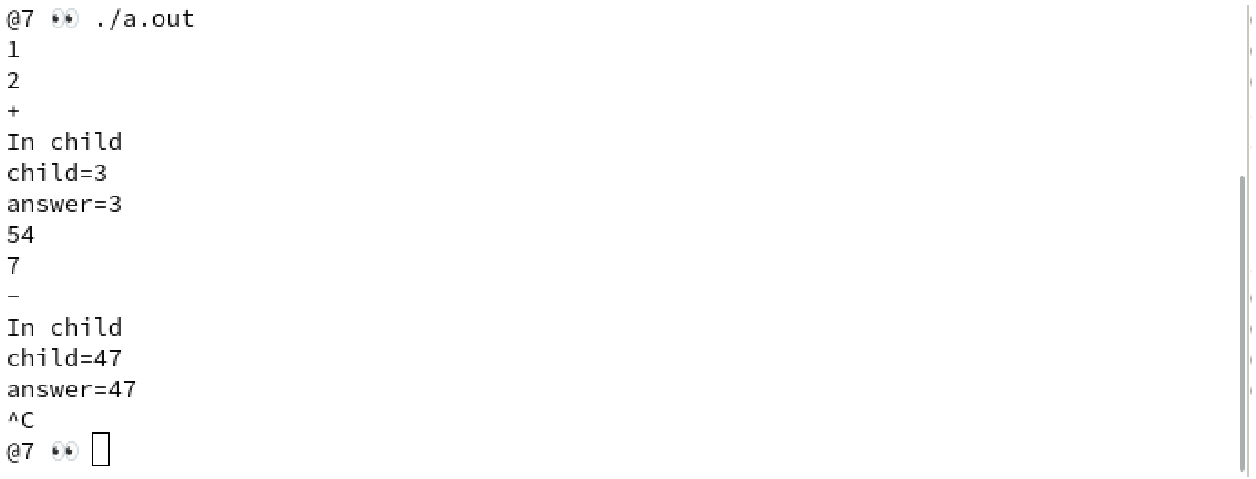
client.c file:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>

int main(){
  char aa[5];
  char bb[5];
  pid_t child;
  int status;

  int a, b;
  char op[5];;

  while (1) {
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%s", op);

    sprintf(aa, "%d", a);
    sprintf(bb, "%d", b);

    child = fork();
    if(child==0)
      execl("./server.out","server",aa,bb,op,(char *)NULL);

    wait(&status);

    printf("answer=%d\n", (int)WEXITSTATUS(status));
  }
}
```

server.c file:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>

int main(int argc,char *argv[]){
  printf("In child\n");

  int a = atoi(argv[1]);
  int b = atoi(argv[2]);

  if(argv[3][0] == '+') {
    printf("child=%d\n",a+b);
    exit(a+b);
  }
  else {
    printf("child=%d\n",a-b);
    exit(a-b);
  }
}
```

Output:



```
@7 👀 ./a.out
1
2
+
In child
child=3
answer=3
54
7
-
In child
child=47
answer=47
^C
@7 👀
```

8. Write a C program that takes a file name as a command line parameter and sorts a set of integers stored in the file (use any sorting method). You can assume that the file will always be there in the current directory and that it will always contain a set of integers (maximum no. of integers is 1000). The sorted output is written to the display and the input file is left unchanged. Compile the C file into an executable named “sort1”. Name the C file sort1.c. Now write a C program (xsort.c) that implements a command called “xsort” that you will invoke from the shell prompt. The syntax of the command is “xsort <filename>”. When you type the command, the command opens a new xterm window (terminal), and then sorts the integers stored in the file <filename> using the program “sort1”. Look up the man pages for xterm, fork and the different variations of exec* system calls (such as execv, execve, execlp etc.) to do this assignment. Submit the C files sort1.c and xsort.c.

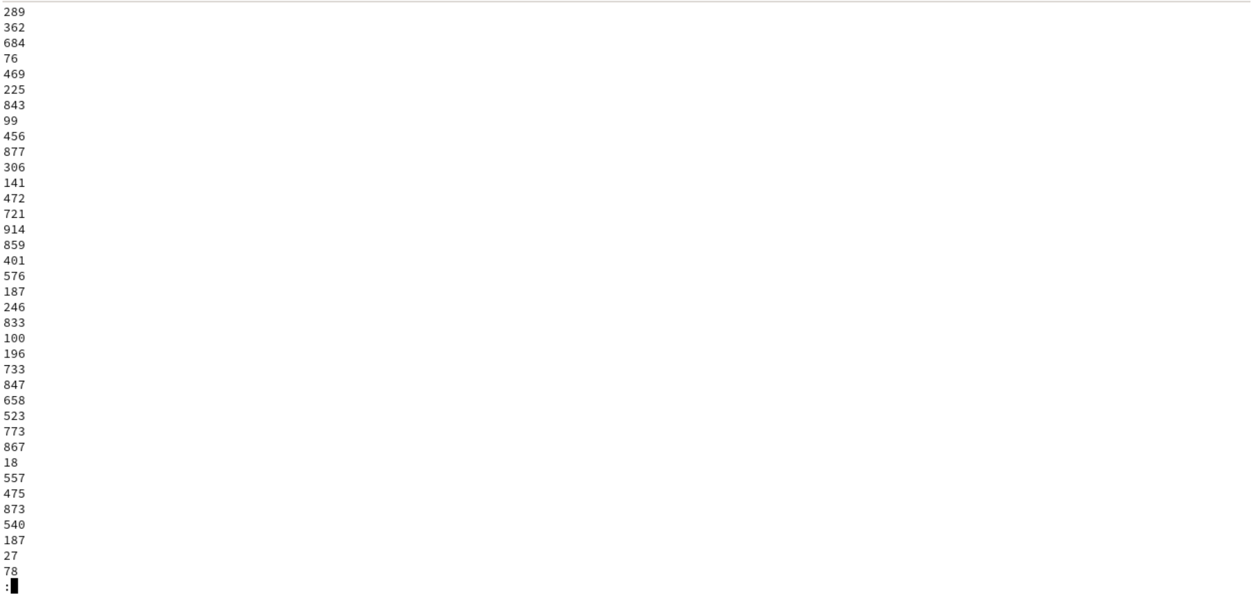
sort1.c file:

```
#include <stdio.h>
FILE *fp;
const char EOL = '\n';

void insertionSort(int array[], int size) {
  for (int step = 1; step < size; step++) {
    int key = array[step];
    int j = step - 1;

    while (key < array[j] && j >= 0) {
      array[j + 1] = array[j];
      --j;
    }
    array[j + 1] = key;
  }
}

int main(int argc, char** argv)
{
    fp = fopen(argv[1], "r");
    char buffer[4];
    int arr[1000];
    int i = 0;
    freopen(argv[1],"r",stdin);
    fclose(fp);

    while(scanf("%i",&arr[i])==1 && buffer[i] != EOF)
        ++i;

    insertionSort(arr, i);
    fp = fopen(argv[1], "w");
    for(int j=0; j<i; ++j)
        fprintf(fp, "%d\n",arr[j]);

    fclose(stdin);

    return 0;
}
```

xsort.c file:

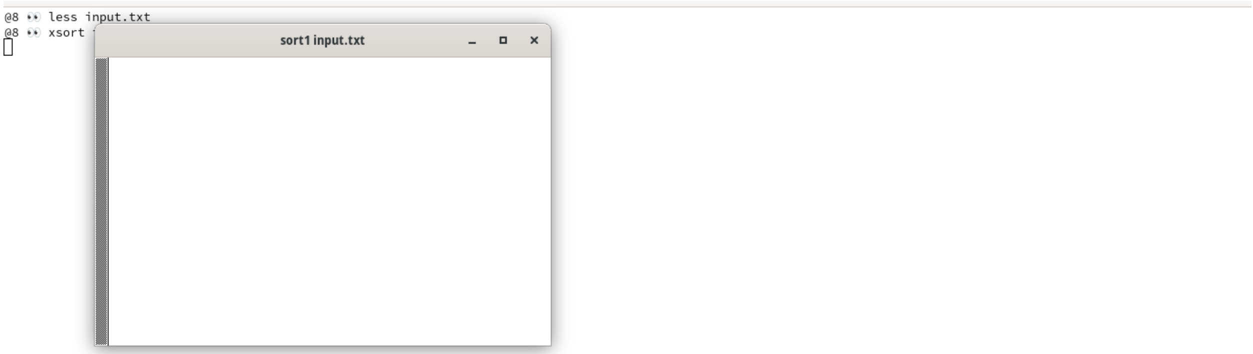
```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>
#include <string.h>

void concat(char* s1, char* s2, char* ns) {
    strcpy(ns, s1);
    strcat(ns, s2);
}

int main(int argc, char** argv){

    char* s1 = "./sort1 ";
    char* s2 = argv[1];
    char s3[strlen(s1) + strlen(s2)];
    concat(s1, s2, s3);

    execl("/usr/bin/xterm", "xterm", "-hold",  "-e", s3,  NULL);

  return 0;
}
```

Below is the contents of **input.txt** file which contains a list of unsorted numbers.



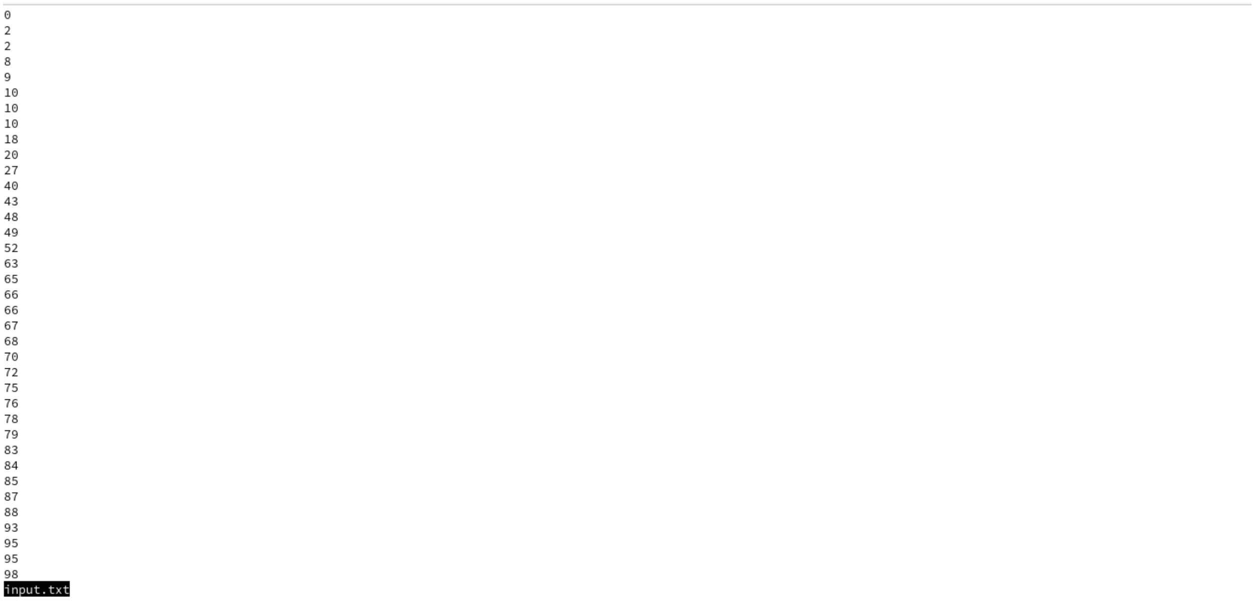
```
289
362
684
76
469
225
843
99
456
877
306
141
472
721
914
859
401
576
187
246
833
100
196
733
847
658
523
773
867
18
557
475
873
540
187
27
78
:
```

Now, we run the newly implemented xsort command and pass the name of input file(input.txt) as a command line argument.



```
@8 👀 less input.txt
@8 👀 xsort 
```

Then, we check the input.txt file contents to know whether the numbers have been sorted or not.



```
0
2
2
8
9
10
10
10
18
20
27
40
43
48
49
52
63
65
66
66
67
68
70
72
75
76
78
79
83
84
85
87
88
93
95
95
98
input.txt
```

It is clear that the numbers have been sorted.

9. Perform a parallel matrix multiplication using POSIX threads in C/C++. Compare the execution time of the program against serial matrix multiplication code by modifying the number of threads used to do the matrix multiplication.

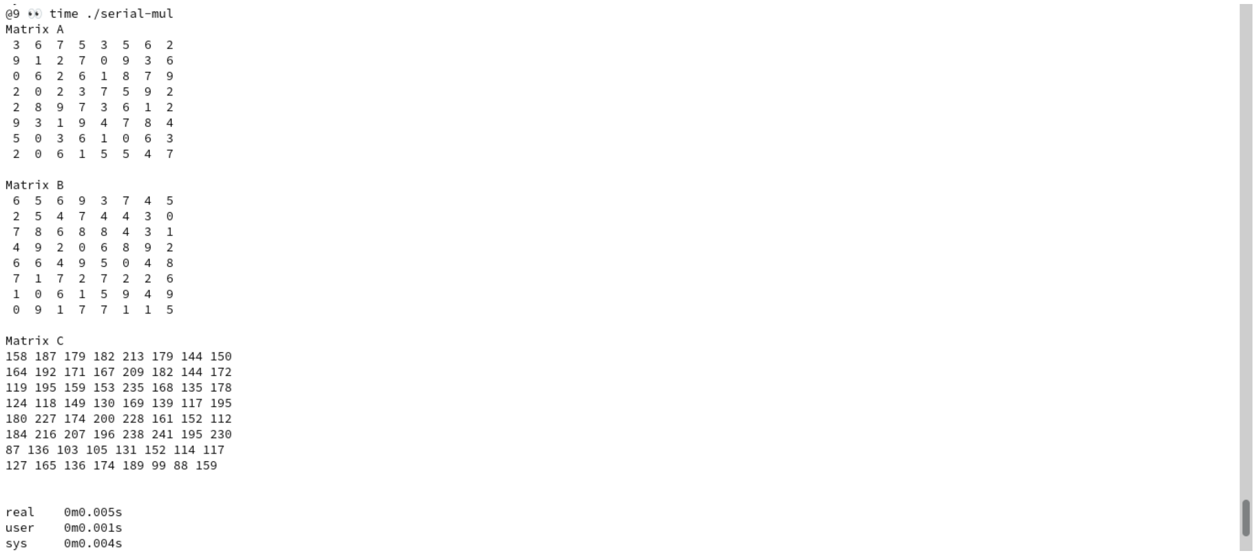
First, we will write a simple serial matrix multiplication program that finds the resultant matrix using the “usual” matrix multiplication. Then, we will write a threaded version of the same program in which each thread will calculate one row of the resultant matrix.

Since, in my machine, there are 4 CPU cores and each is capable of handling 2 threads per core, I’m using two 8 x 8 matrices to do the matrix multiplication.

serial-mul.cpp file:

```
#include <iomanip>
#include "iostream"
#include "vector"

using namespace std;

#define MAX_SIZE 8

// function to initialize the matrix with random integer values between 0 - MAX_SIZE *
MAX_SIZE
void initializeRandomMatrix(vector<vector<int>>& mat, int size) {
    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
            mat[i][j] = random() % (10);
}

// function to print the matrix
void printMatrix(vector<vector<int>>& mat) {
    for (int i = 0; i < mat.size(); i++) {
        for (int j = 0; j < mat[0].size(); j++)
            cout << setw(2) << mat[i][j] << ' ';
        cout << endl;
    }
    cout << endl;
}

int main() {
    // create two matrices matA, matB of order MAX_SIZE
    vector<vector<int>> matA(MAX_SIZE, vector<int>(MAX_SIZE));
    vector<vector<int>> matB(MAX_SIZE, vector<int>(MAX_SIZE));
    vector<vector<int>> matC(MAX_SIZE, vector<int>(MAX_SIZE));
    // initialize both matrices with random integer values
    initializeRandomMatrix(matA, MAX_SIZE);
    initializeRandomMatrix(matB, MAX_SIZE);

    // print both matrices
    cout << "Matrix A" << endl;
    printMatrix(matA);
    cout << "Matrix B" << endl;
    printMatrix(matB);

    // do serial multiplication
    for (int i = 0; i < MAX_SIZE; i++)
        for (int j = 0; j < MAX_SIZE; j++)
            for (int k = 0; k < MAX_SIZE; k++)
                matC[i][j] += matA[i][k] * matB[k][j];

    // print the resultant matrix
    cout << "Matrix C" << endl;
    printMatrix(matC);

    return 0;
}
```

Its output, along with the time usage is as follows.



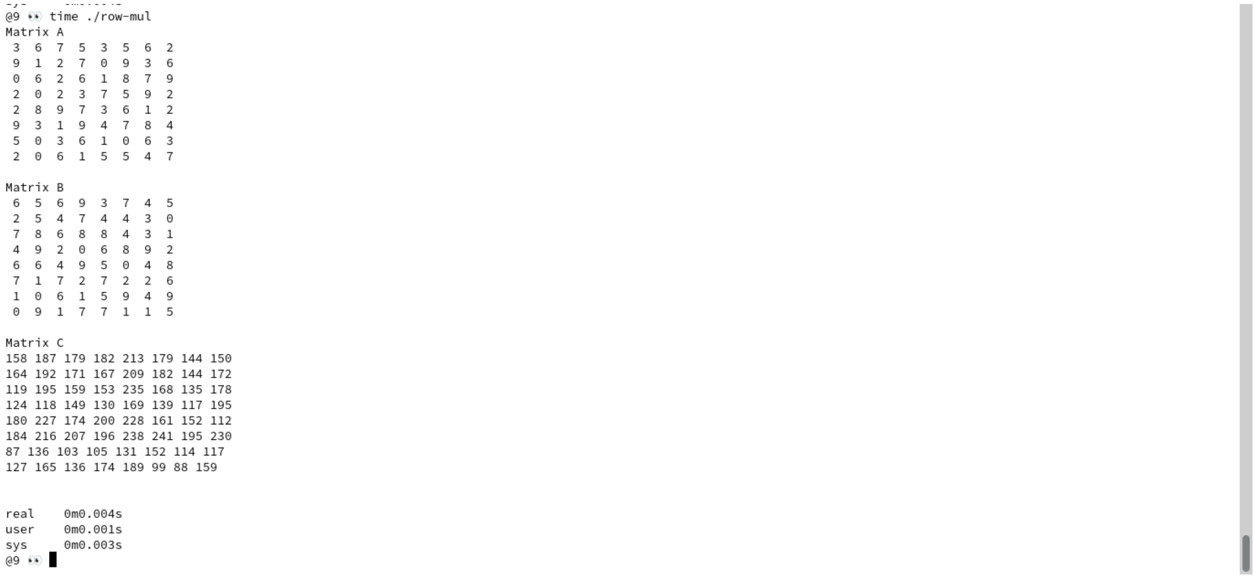
```
@9 👀 time ./serial-mul
Matrix A
 3  6  7  5  3  5  6  2
 9  1  2  7  0  9  3  6
 0  6  2  6  1  8  7  9
 2  0  2  3  7  5  9  2
 2  8  9  7  3  6  1  2
 9  3  1  9  4  7  8  4
 5  0  3  6  1  0  6  3
 2  0  6  1  5  5  4  7

Matrix B
 6  5  6  9  3  7  4  5
 2  5  4  7  4  4  3  0
 7  8  6  8  8  4  3  1
 4  9  2  0  6  8  9  2
 6  6  4  9  5  0  4  8
 7  1  7  2  7  2  2  6
 1  0  6  1  5  9  4  9
 0  9  1  7  7  1  1  5

Matrix C
158 187 179 182 213 179 144 150
164 192 171 167 209 182 144 172
119 195 159 153 235 168 135 178
124 118 149 130 169 139 117 195
180 227 174 200 228 161 152 112
184 216 207 196 238 241 195 230
87 136 103 105 131 152 114 117
127 165 136 174 189 99 88 159


real    0m0.005s
user    0m0.001s
sys     0m0.004s
```

row-mul.cpp file(threaded version):

```
#include <iomanip>
```

```
#include "iostream"
#include "vector"

using namespace std;

#define MAX_THREAD 8
#define MAX_SIZE 8

// create two matrices matA, matB of order MAX_SIZE
vector<vector<int>> matA(MAX_SIZE, vector<int>(MAX_SIZE));
vector<vector<int>> matB(MAX_SIZE, vector<int>(MAX_SIZE));
vector<vector<int>> matC(MAX_SIZE, vector<int>(MAX_SIZE));

// variable to track the row number calculated
int rowCounter = 0;

// function to initialize the matrix with random integer values between 0 - MAX_SIZE *
MAX_SIZE
void initializeRandomMatrix(vector<vector<int>>& mat, int size) {
    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
            mat[i][j] = random() % (10);
}

// function to print the matrix
void printMatrix(vector<vector<int>>& mat) {
    for (int i = 0; i < mat.size(); i++) {
        for (int j = 0; j < mat[0].size(); j++)
            cout << setw(2) << mat[i][j] << ' ';
        cout << endl;
    }
    cout << endl;
}

// thread routine to calculate the row of the product matrix using corresponding row of first matrix
// and corresponding column of second matrix
void* calculateRow(void* arg) {
    int rowNumber = rowCounter++;

    for (int i = 0; i < MAX_SIZE; i++)
        for (int j = 0; j < MAX_SIZE; j++)
            matC[rowNumber][i] += matA[rowNumber][j] * matB[j][i];

    return nullptr;
}



int main() {
    // initialize both matrices with random integer values
    initializeRandomMatrix(matA, MAX_SIZE);
    initializeRandomMatrix(matB, MAX_SIZE);

    // print both matrices
    cout << "Matrix A" << endl;
    printMatrix(matA);
    cout << "Matrix B" << endl;
    printMatrix(matB);

    // create one thread per row and start evaluating corresponding row of resultant matrix
    pthread_t threads[MAX_SIZE];
    for (int i = 0; i < MAX_SIZE; i++) {
        int* p;
        pthread_create(&threads[i], NULL, calculateRow, (void*)p);
    }

    // wait for all threads to complete
    for (int i = 0; i < MAX_SIZE; i++)
        pthread_join(threads[i], NULL);

    // print the resultant matrix
    cout << "Matrix C" << endl;
    printMatrix(matC);

    return 0;
}
```

Its output, along with the time usage is as follows.



```
@9 👀 time ./row-mul
Matrix A
 3  6  7  5  3  5  6  2
 9  1  2  7  0  9  3  6
 0  6  2  6  1  8  7  9
 2  0  2  3  7  5  9  2
 2  8  9  7  3  6  1  2
 9  3  1  9  4  7  8  4
 5  0  3  6  1  0  6  3
 2  0  6  1  5  5  4  7

Matrix B
 6  5  6  9  3  7  4  5
 2  5  4  7  4  4  3  0
 7  8  6  8  8  4  3  1
 4  9  2  0  6  8  9  2
 6  6  4  9  5  0  4  8
 7  1  7  2  7  2  2  6
 1  0  6  1  5  9  4  9
 0  9  1  7  7  1  1  5

Matrix C
158 187 179 182 213 179 144 150
164 192 171 167 209 182 144 172
119 195 159 153 235 168 135 178
124 118 149 130 169 139 117 195
180 227 174 200 228 161 152 112
184 216 207 196 238 241 195 230
87 136 103 105 131 152 114 117
127 165 136 174 189 99 88 159


real    0m0.004s
user    0m0.001s
sys     0m0.003s
@9 👀 █
```

As, we can see from the output, the threaded version(0.004s) is slightly faster than the serial version(0.005s).

However, it should be noted that by just increasing the number of threads, the execution time won’t increase as creating and managing them is also an overhead for the CPU. Threads are best utilized when one thread is used per CPU(in a multiprocessor environment) or per core of CPU(in a uniprocessor environment with multiple cores).