SOFTSERVE PYTHON DEVELOPER INTERNSHIP REPORT ON LINUX OPERATING SYSTEM

Report Number 3

“Process in Linux Operating system”

Intern name: Bahamboula Idourah C. Y.

City: Kharkov Group: Kh-064-Python

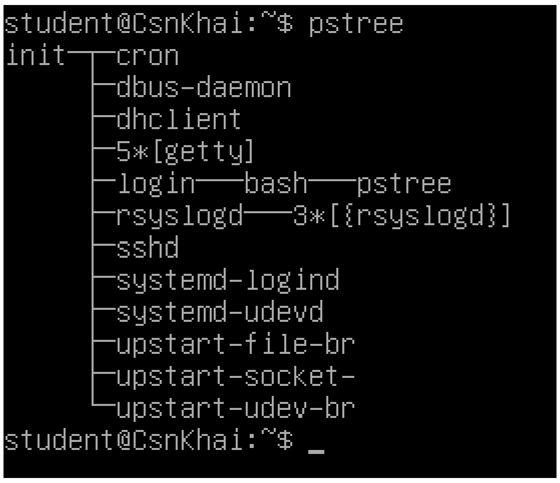
Instructor: Дмитрий Узун

Kharkov

Task 1

1) How many states could has a process in Linux
   - Running
   - Sleeping
   - Stopped
   - Zombie

2) Examine the pstree command. Make output (highlight) the chain (ancestors) of the current process.



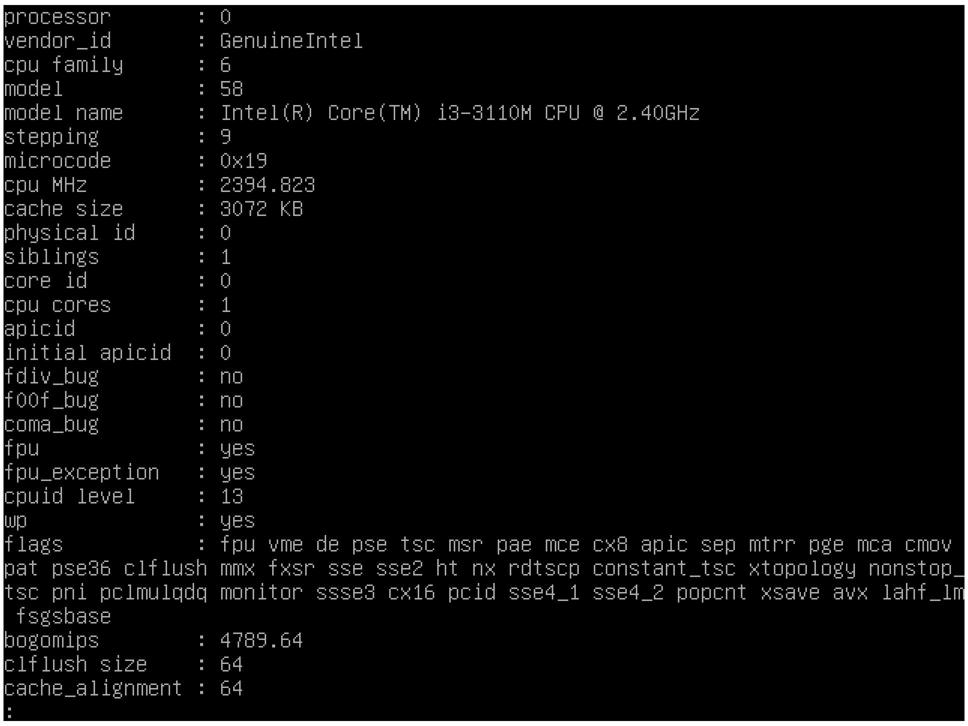
```
student@CsnKhai:~$ pstree
init─┬─cron
     ├─dbus-daemon
     ├─dhclient
     ├─5*[getty]
     ├─login───bash───pstree
     ├─rsyslogd───3*[{rsyslogd}]
     ├─sshd
     ├─systemd-logind
     ├─systemd-udevd
     ├─upstart-file-br
     ├─upstart-socket-
     └─upstart-udev-br
student@CsnKhai:~$ _
```

Init ---> login ---> bash---> pstree

3) What is a proc file system?

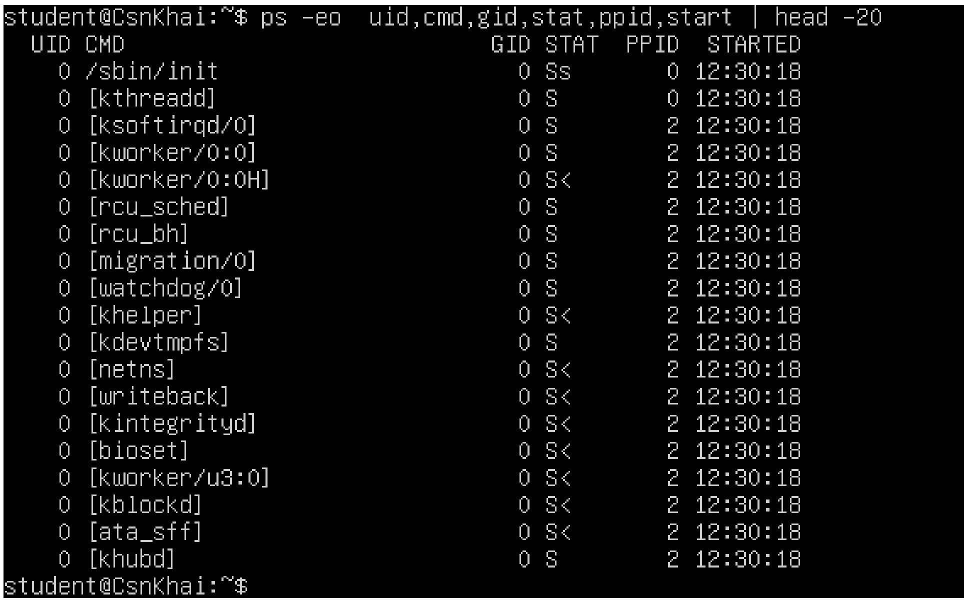
Proc is a system file which resides in the computer memory and provide important information about the system status such as information related to processes, devices, kernel, I/O, interrupts.

4) Print information about the processor (its type, supported technologies, etc.).



```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz
stepping        : 9
microcode       : 0x19
cpu MHz         : 2394.823
cache size      : 3072 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fdiv_bug        : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ht nx rdtscp constant_tsc xtopology nonstop_
tsc pni pclmulqdq monitor ssse3 cx16 pcid sse4_1 sse4_2 popcnt xsave avx lahf_lm
 fsgsbase
bogomips        : 4789.64
clflush size    : 64
cache_alignment : 64
:
```

5) Use the ps command to get information about the process. The information should be as follows: the owner of the process, the arguments with which the process was launched for execution, the group owner of this process, etc



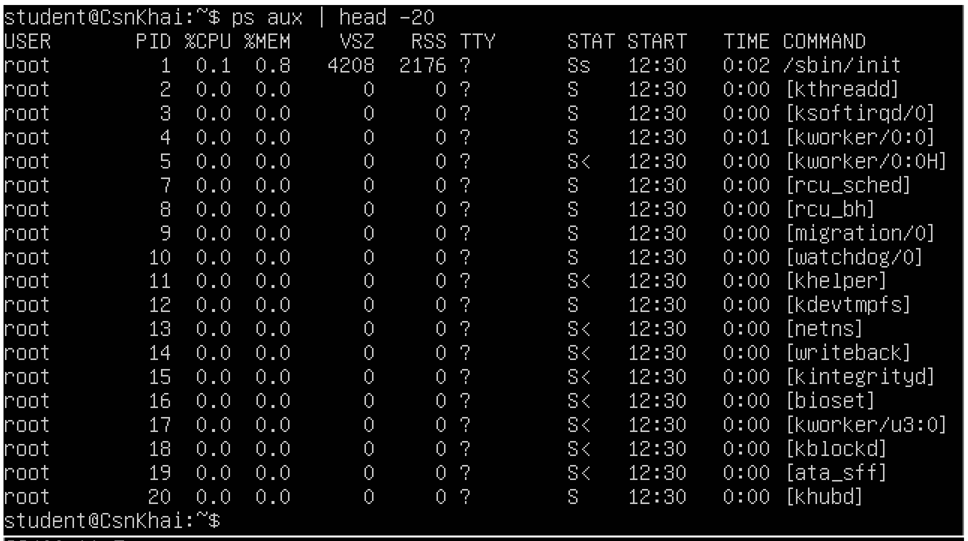
```
student@CsnKhai:~$ ps -eo  uid,cmd,gid,stat,ppid,start | head -20
  UID CMD                          GID STAT  PPID  STARTED
    0 /sbin/init                     0 Ss       0 12:30:18
    0 [kthreadd]                     0 S        0 12:30:18
    0 [ksoftirqd/0]                  0 S        2 12:30:18
    0 [kworker/0:0]                  0 S        2 12:30:18
    0 [kworker/0:0H]                 0 S<       2 12:30:18
    0 [rcu_sched]                    0 S        2 12:30:18
    0 [rcu_bh]                       0 S        2 12:30:18
    0 [migration/0]                  0 S        2 12:30:18
    0 [watchdog/0]                   0 S        2 12:30:18
    0 [khelper]                      0 S<       2 12:30:18
    0 [kdevtmpfs]                    0 S        2 12:30:18
    0 [netns]                        0 S<       2 12:30:18
    0 [writeback]                    0 S<       2 12:30:18
    0 [kintegrityd]                  0 S<       2 12:30:18
    0 [bioset]                       0 S<       2 12:30:18
    0 [kworker/u3:0]                 0 S<       2 12:30:18
    0 [kblockd]                      0 S<       2 12:30:18
    0 [ata_sff]                      0 S<       2 12:30:18
    0 [khubd]                        0 S        2 12:30:18
student@CsnKhai:~$
```

6) How to define kernel processes and user processes?

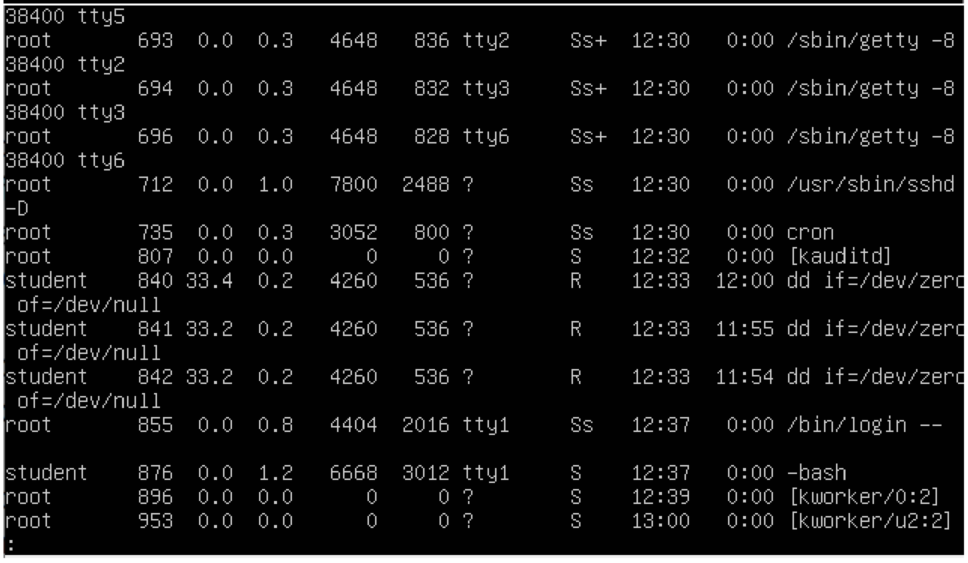
Kernel processes are part of the Linux kernel and their priority cannot be adjusted, they cannot be killed unless taking the machine down.

User process are process initiated by the user the can be killed, stopped or changed priority.

7) Print the list of processes to the terminal. Briefly describe the statuses of the processes. What condition are they in, or can they be arriving in?



```
student@CsnKhai:~$ ps aux | head -20
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.8   4208  2176 ?        Ss   12:30   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    12:30   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    12:30   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    12:30   0:01 [kworker/0:0]
root         5  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    12:30   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    12:30   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    12:30   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    12:30   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S<   12:30   0:00 [khelper]
root        12  0.0  0.0      0     0 ?        S    12:30   0:00 [kdevtmpfs]
root        13  0.0  0.0      0     0 ?        S<   12:30   0:00 [netns]
root        14  0.0  0.0      0     0 ?        S<   12:30   0:00 [writeback]
root        15  0.0  0.0      0     0 ?        S<   12:30   0:00 [kintegrityd]
root        16  0.0  0.0      0     0 ?        S<   12:30   0:00 [bioset]
root        17  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/u3:0]
root        18  0.0  0.0      0     0 ?        S<   12:30   0:00 [kblockd]
root        19  0.0  0.0      0     0 ?        S<   12:30   0:00 [ata_sff]
root        20  0.0  0.0      0     0 ?        S    12:30   0:00 [khubd]
student@CsnKhai:~$
```



```
38400 tty5
root       693  0.0  0.3   4648   836 tty2     Ss+  12:30   0:00 /sbin/getty -8
38400 tty2
root       694  0.0  0.3   4648   832 tty3     Ss+  12:30   0:00 /sbin/getty -8
38400 tty3
root       696  0.0  0.3   4648   828 tty6     Ss+  12:30   0:00 /sbin/getty -8
38400 tty6
root       712  0.0  1.0   7800  2488 ?        Ss   12:30   0:00 /usr/sbin/sshd
-D
root       735  0.0  0.3   3052   800 ?        Ss   12:30   0:00 cron
root       807  0.0  0.0      0     0 ?        S    12:32   0:00 [kauditd]
student    840 33.4  0.2   4260   536 ?        R    12:33  12:00 dd if=/dev/zero
 of=/dev/null
student    841 33.2  0.2   4260   536 ?        R    12:33  11:55 dd if=/dev/zero
 of=/dev/null
student    842 33.2  0.2   4260   536 ?        R    12:33  11:54 dd if=/dev/zero
 of=/dev/null
root       855  0.0  0.8   4404  2016 tty1     Ss   12:37   0:00 /bin/login --

student    876  0.0  1.2   6668  3012 tty1     S    12:37   0:00 -bash
root       896  0.0  0.0      0     0 ?        S    12:39   0:00 [kworker/0:2]
root       953  0.0  0.0      0     0 ?        S    13:00   0:00 [kworker/u2:2]
:
```

(S) Sleeping: when waiting for an event to complete

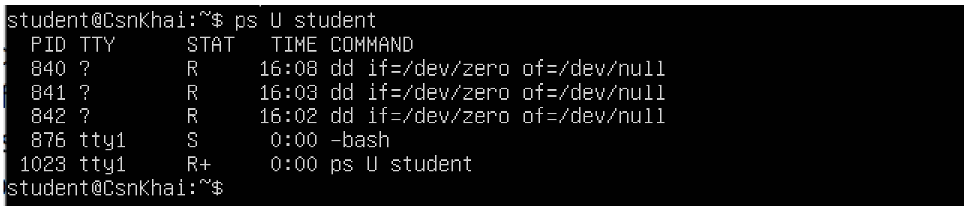
(R) Running: currently active and is using the CPU time

(D) Uninterrupted: Sleep state that cannot be stopped. Usually when waiting for I/O

(Z) Zombie: the program has been stopped but could been removed by its parent

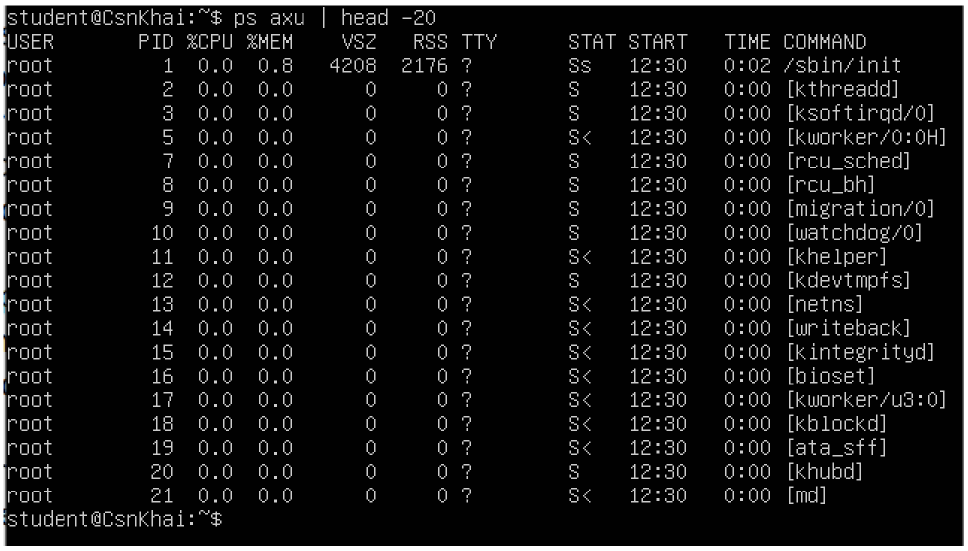
Process 841 and 842 Created by user student are Running and most of process are sleeping.

8) Display only the processes of a specific user.



```
student@CsnKhai:~$ ps U student
  PID TTY      STAT   TIME COMMAND
  840 ?        R     16:08 dd if=/dev/zero of=/dev/null
  841 ?        R     16:03 dd if=/dev/zero of=/dev/null
  842 ?        R     16:02 dd if=/dev/zero of=/dev/null
  876 tty1     S      0:00 -bash
 1023 tty1     R+     0:00 ps U student
student@CsnKhai:~$
```

9) What utilities can be used to analyze existing running tasks (by analyzing the help for the ps command)



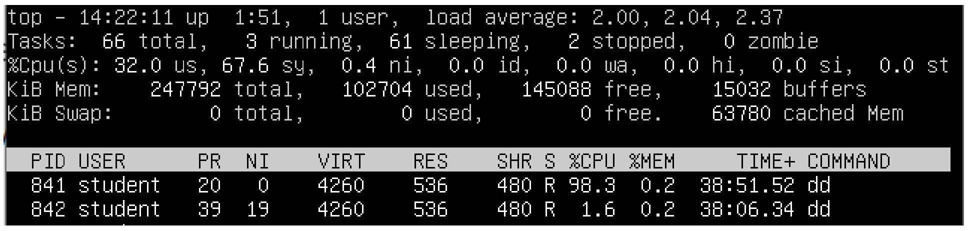
```
student@CsnKhai:~$ ps axu | head -20
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.8   4208  2176 ?        Ss   12:30   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    12:30   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    12:30   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    12:30   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    12:30   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    12:30   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    12:30   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S<   12:30   0:00 [khelper]
root        12  0.0  0.0      0     0 ?        S    12:30   0:00 [kdevtmpfs]
root        13  0.0  0.0      0     0 ?        S<   12:30   0:00 [netns]
root        14  0.0  0.0      0     0 ?        S<   12:30   0:00 [writeback]
root        15  0.0  0.0      0     0 ?        S<   12:30   0:00 [kintegrityd]
root        16  0.0  0.0      0     0 ?        S<   12:30   0:00 [bioset]
root        17  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/u3:0]
root        18  0.0  0.0      0     0 ?        S<   12:30   0:00 [kblockd]
root        19  0.0  0.0      0     0 ?        S<   12:30   0:00 [ata_sff]
root        20  0.0  0.0      0     0 ?        S    12:30   0:00 [khubd]
root        21  0.0  0.0      0     0 ?        S<   12:30   0:00 [md]
student@CsnKhai:~$
```

10) What information does top command display?

top command is used to show the Linux processes. It provides a dynamic real-time view of the running system.

11) Display the processes of the specific user using the top command.

Command top U student or top U [username]

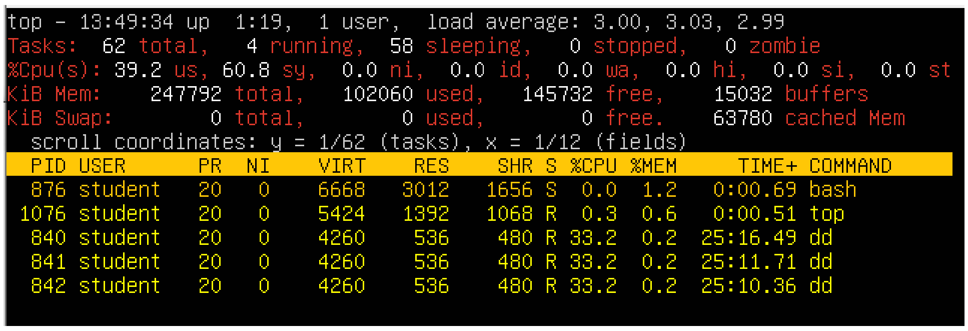


```
top - 14:22:11 up  1:51,  1 user,  load average: 2.00, 2.04, 2.37
Tasks:  66 total,   3 running,  61 sleeping,   2 stopped,   0 zombie
%Cpu(s): 32.0 us, 67.6 sy,  0.4 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   102704 used,   145088 free,    15032 buffers
KiB Swap:        0 total,        0 used,        0 free.    63780 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  841 student   20   0    4260    536    480 R  98.3  0.2  38:51.52 dd
  842 student   39  19    4260    536    480 R   1.6  0.2  38:06.34 dd
```

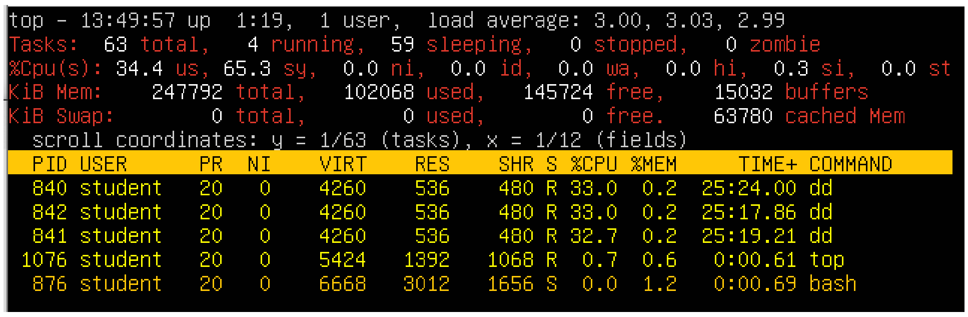
12) What interactive commands can be used to control the top command? Give a couple of examples.

Shift + Z to change the color



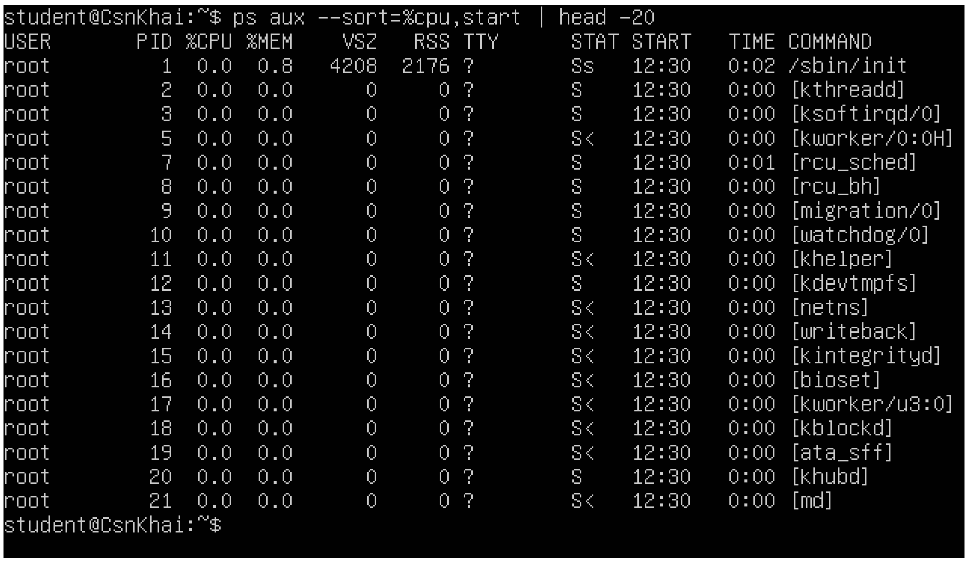
```
top - 13:49:34 up  1:19,  1 user,  load average: 3.00, 3.03, 2.99
Tasks:  62 total,   4 running,  58 sleeping,   0 stopped,   0 zombie
%Cpu(s): 39.2 us, 60.8 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   102060 used,   145732 free,    15032 buffers
KiB Swap:        0 total,        0 used,        0 free.    63780 cached Mem
  scroll coordinates: y = 1/62 (tasks), x = 1/12 (fields)
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  876 student   20   0    6668   3012   1656 S   0.0  1.2   0:00.69 bash
 1076 student   20   0    5424   1392   1068 R   0.3  0.6   0:00.51 top
  840 student   20   0    4260    536    480 R  33.2  0.2  25:16.49 dd
  841 student   20   0    4260    536    480 R  33.2  0.2  25:11.71 dd
  842 student   20   0    4260    536    480 R  33.2  0.2  25:10.36 dd
```

Shift + R to Reverse sort



```
top - 13:49:57 up  1:19,  1 user,  load average: 3.00, 3.03, 2.99
Tasks:  63 total,   4 running,  59 sleeping,   0 stopped,   0 zombie
%Cpu(s): 34.4 us, 65.3 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem:    247792 total,   102068 used,   145724 free,    15032 buffers
KiB Swap:        0 total,        0 used,        0 free.    63780 cached Mem
  scroll coordinates: y = 1/63 (tasks), x = 1/12 (fields)
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  840 student   20   0    4260    536    480 R 33.0  0.2  25:24.00 dd
  842 student   20   0    4260    536    480 R 33.0  0.2  25:17.86 dd
  841 student   20   0    4260    536    480 R 32.7  0.2  25:19.21 dd
 1076 student   20   0    5424   1392   1068 R  0.7  0.6   0:00.61 top
  876 student   20   0    6668   3012   1656 S  0.0  1.2   0:00.69 bash
```

13) Sort the contents of the processes window using various parameters (for example, the amount of processor time taken up, etc.)



```
student@CsnKhai:~$ ps aux --sort=%cpu,start | head -20
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.8   4208  2176 ?        Ss   12:30   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    12:30   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    12:30   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    12:30   0:01 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    12:30   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    12:30   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    12:30   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S<   12:30   0:00 [khelper]
root        12  0.0  0.0      0     0 ?        S    12:30   0:00 [kdevtmpfs]
root        13  0.0  0.0      0     0 ?        S<   12:30   0:00 [netns]
root        14  0.0  0.0      0     0 ?        S<   12:30   0:00 [writeback]
root        15  0.0  0.0      0     0 ?        S<   12:30   0:00 [kintegrityd]
root        16  0.0  0.0      0     0 ?        S<   12:30   0:00 [bioset]
root        17  0.0  0.0      0     0 ?        S<   12:30   0:00 [kworker/u3:0]
root        18  0.0  0.0      0     0 ?        S<   12:30   0:00 [kblockd]
root        19  0.0  0.0      0     0 ?        S<   12:30   0:00 [ata_sff]
root        20  0.0  0.0      0     0 ?        S    12:30   0:00 [khubd]
root        21  0.0  0.0      0     0 ?        S<   12:30   0:00 [md]
student@CsnKhai:~$
```

14) . Concept of priority, what commands are used to set priority?

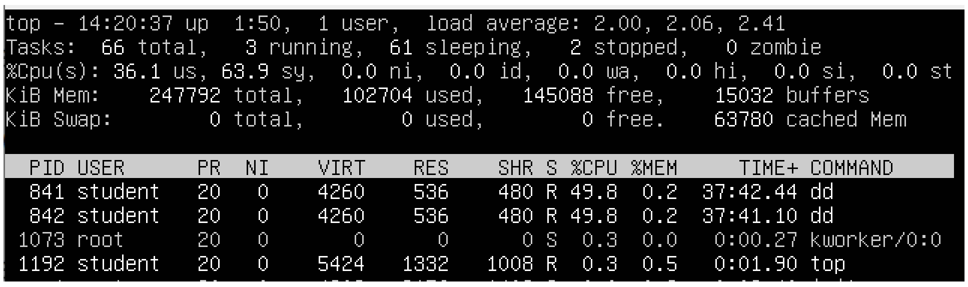
Priority of process can be viewed by examining PRI column of ps –l command result or top. Higher is the priority, more CPU time is allowed to the process and vice versa. If authorized a user can set priority of a process using command nice [-n value] [command [argument..]] or using top command by press r and providing the process PID then the nice value.

The value of nice ranges from –20 to +19.

15) Can I change the priority of a process using the top command? If so, how?

Yes the priority of a process can be changed using top command typing r then giving the PID and nice value

PID = 842 NI = 0 before being changed (see Fig 15.1)

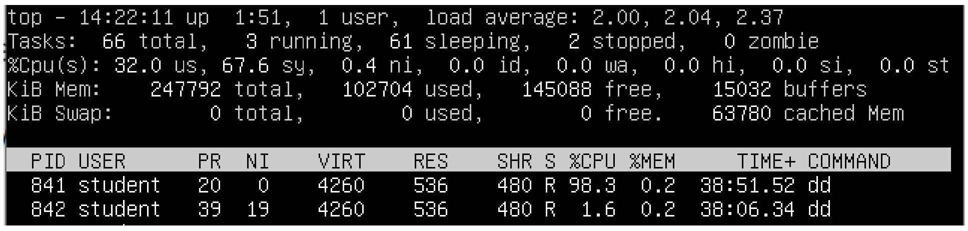


```
top - 14:20:37 up  1:50,  1 user,  load average: 2.00, 2.06, 2.41
Tasks:  66 total,   3 running,  61 sleeping,   2 stopped,   0 zombie
%Cpu(s): 36.1 us, 63.9 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   102704 used,   145088 free,    15032 buffers
KiB Swap:        0 total,        0 used,        0 free.    63780 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  841 student   20   0    4260    536    480 R 49.8  0.2  37:42.44 dd
  842 student   20   0    4260    536    480 R 49.8  0.2  37:41.10 dd
 1073 root      20   0       0      0      0 S  0.3  0.0   0:00.27 kworker/0:0
 1192 student   20   0    5424   1332   1008 R  0.3  0.5   0:01.90 top
```

Fig 15.1

PID = 842 NI = 19 after being changed (see Fig 15.2)

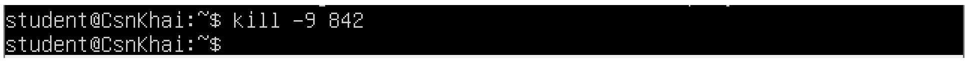


```
top - 14:22:11 up  1:51,  1 user,  load average: 2.00, 2.04, 2.37
Tasks:  66 total,   3 running,  61 sleeping,   2 stopped,   0 zombie
%Cpu(s): 32.0 us, 67.6 sy,  0.4 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   102704 used,   145088 free,    15032 buffers
KiB Swap:        0 total,        0 used,        0 free.    63780 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  841 student   20   0    4260    536    480 R 98.3  0.2  38:51.52 dd
  842 student   39  19    4260    536    480 R  1.6  0.2  38:06.34 dd
```

Fig 15.2

16) Examine the kill command. How to send with the kill command process control signal? Give an example of commonly used signals.



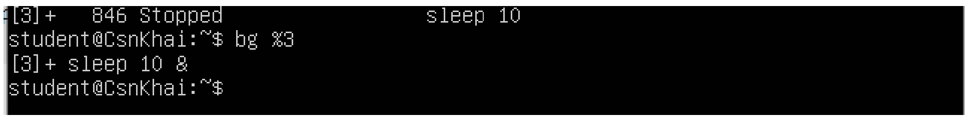
```
student@CsnKhai:~$ kill -9 842
student@CsnKhai:~$
```

- 1 (SIGHUP) send by a parent process to all its child when it’s stopped
- 2 (SIGNINT) equivalents to <Ctrl> + <C>;
- 9 (SIGKILL) stop the process
- 15 (SIGTERM) the default system of kill command.

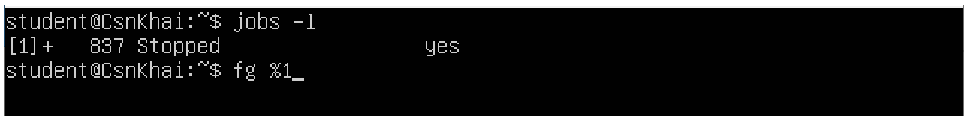
17) Commands jobs, fg, bg, nohup. What are they for? Use the sleep, yes command to demonstrate the process control mechanism with fg, bg.

jobs, fg, bg nohup are command to interact with process in Linux

- jobs: list jobs in current shell;
- bg: sends process in the background;
- fg: brings process to foreground;
- nohup: is used to start a process which will still running event when the user who initiated it will log out the system.



```
[3]+   846 Stopped                 sleep 10
student@CsnKhai:~$ bg %3
[3]+ sleep 10 &
student@CsnKhai:~$
```



```
student@CsnKhai:~$ jobs -l
[1]+   837 Stopped                 yes
student@CsnKhai:~$ fg %1_
```