Part1

1. How many states could has a process in Linux?

8 states: Running, Sleeping (Interruptible), Sleeping (Uninterruptible), Stopped, Zombie, Traced, Paging, Dead.

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

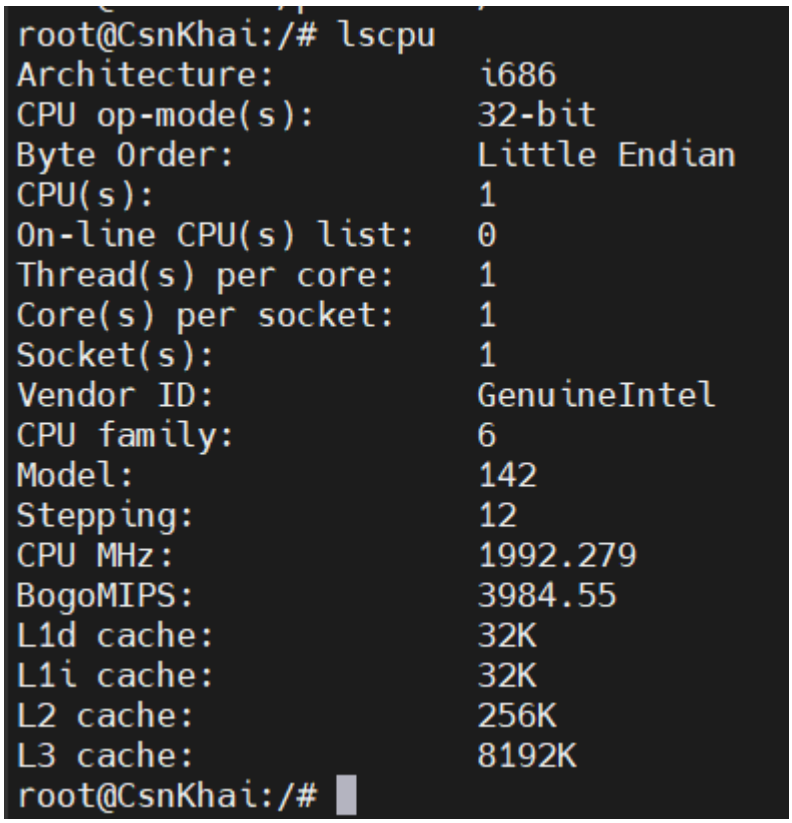
```
root@CsnKhai:~# pstree -a
init
  ├─cron
  ├─dbus-daemon --system --fork
  ├─dhclient -1 -v -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases eth0
  ├─getty -8 38400 tty4
  ├─getty -8 38400 tty5
  ├─getty -8 38400 tty2
  ├─getty -8 38400 tty3
  ├─getty -8 38400 tty6
  ├─login --
  │   └─bash
  ├─rsyslogd
  │   └─3*[{rsyslogd}]
  ├─sshd -D
  │   ├─sshd
  │   │   └─sshd
  │   │       └─bash
  │   │           └─sudo su
  │   │               └─su
  │   │                   └─bash
  │   │                       └─pstree -a
  │   └─sshd
  │       └─sshd
  │           └─sftp-server
  ├─systemd-logind
  ├─systemd-udevd --daemon
  ├─upstart-file-br --daemon
  ├─upstart-socket- --daemon
  └─upstart-udev-br --daemon
root@CsnKhai:~#
```

3. What is a proc file system?

The /proc file system in Linux is a virtual file system that provides an interface to kernel data structures and information. It allows processes and programs to access and manipulate kernel-related information as if it were a file system. The /proc file system doesn't store real files on disk; instead, it provides a way to interact with kernel data and configuration through a file-like interface.

```
root@CsnKhai:/proc# ls
1     12   19   279  4    69   8    917       cgroups     execdomains  kallsyms     locks        partitions  sys              vmallocinfo
10    126  2    28   42   7    820  918       cmdline     fb           kcore        mdstat       sched_debug sysrq-trigger    vmstat
11    127  20   29   44   719  833  921       consoles    filesystems  keys         meminfo      schedstat   sysvipc          zoneinfo
1123  13   21   3    45   721  850  922       cpuinfo     fs           key-users    misc         scsi        timer_list
115   14   22   30   466  724  864  923       crypto      interrupts   kmsg         modules      self        timer_stats
1151  15   25   328  5    725  866  acpi      devices     iomem        kpagecount   mounts       slabinfo    tty
116   16   26   361  56   727  884  asound    diskstats   ioports      kpageflags   mtrr         softirqs    uptime
117   17   27   363  599  751  885  buddyinfo dma         ipmi         latency_stats net         stat        version
1184  18   275  386  68   758  9    bus       driver      irq          loadavg      pagetypeinfo swaps       version_signature
```

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



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.

```
root@CsnKhai:/# ps -eo pid,user,args,group
  PID USER     COMMAND                         GROUP
    1 root     /sbin/init                      root
    2 root     [kthreadd]                      root
    3 root     [ksoftirqd/0]                   root
    4 root     [kworker/0:0]                   root
    5 root     [kworker/0:0H]                  root
    7 root     [rcu_sched]                     root
    8 root     [rcu_bh]                        root
    9 root     [migration/0]                   root
   10 root     [watchdog/0]                    root
   11 root     [khelper]                       root
   12 root     [kdevtmpfs]                     root
   13 root     [netns]                         root
   14 root     [writeback]                     root
   15 root     [kintegrityd]                   root
   16 root     [bioset]                        root
   17 root     [kworker/u3:0]                  root
   18 root     [kblockd]                       root
   19 root     [ata_sff]                       root
   20 root     [khubd]                         root
   21 root     [md]                            root
   22 root     [devfreq_wq]                    root
   25 root     [khungtaskd]                    root
   26 root     [kswapd0]                       root
   27 root     [ksmd]                          root
   28 root     [fsnotify_mark]                 root
   29 root     [ecryptfs-kthrea]               root
   30 root     [crypto]                        root
   42 root     [kthrotld]                      root
   44 root     [scsi_eh_0]                     root
   45 root     [scsi_eh_1]                     root
   56 root     [kworker/0:2]                   root
   68 root     [deferwq]                       root
   69 root     [charger_manager]               root
  115 root     [kpsmoused]                     root
  116 root     [scsi_eh_2]                     root
  117 root     [kworker/u3:1]                  root
  126 root     [jbd2/sda1-8]                   root
```

6. How to define kernel processes and user processes?

Kernel processes:

```
root@CsnKhai:/# ps aux | head
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.8   4196  2180 ?        Ss   17:46   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    17:46   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    17:46   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    17:46   0:00 [kworker/0:0]
root         5  0.0  0.0      0     0 ?        S<   17:46   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    17:46   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    17:46   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    17:46   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    17:46   0:00 [watchdog/0]
root@CsnKhai:/#
```

User processes:

```
root@CsnKhai:/# ps -U student
  PID TTY          TIME CMD
  850 tty1     00:00:00 bash
  884 ?        00:00:00 sshd
  885 pts/0    00:00:00 bash
  917 ?        00:00:00 sshd
  918 ?        00:00:00 sftp-server
```

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?

```
root@CsnKhai:/# ps ef
  PID TTY      STAT   TIME COMMAND
  921 pts/0    S      0:00 sudo su XDG_SESSION_ID=1 TERM=xterm SHELL=/bin/bash SSH_CLIENT=192.168.1.104 63262 22 SSH_TTY=/dev/pts/0 USER=student LS_COLORS=rs=
  922 pts/0    S      0:00  \_ su TERM=xterm LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg
  923 pts/0    S      0:00      \_ bash TERM=xterm LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37
 1231 pts/0    R+     0:00          \_ ps ef XDG_SESSION_ID=1 SHELL=/bin/bash TERM=xterm USER=root LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do
  820 tty1     Ss     0:00 /bin/login --       PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux JOB=rc INSTANCE= RESULT=ok RUNLE
  727 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6 PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux RUNLEVEL=2 PREVLEVEL=N UPSTA
  725 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3 PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux RUNLEVEL=2 PREVLEVEL=N UPSTA
  724 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2 PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux RUNLEVEL=2 PREVLEVEL=N UPSTA
  721 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5 PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux RUNLEVEL=2 PREVLEVEL=N UPSTA
  719 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4 PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin TERM=linux RUNLEVEL=2 PREVLEVEL=N UPSTA
root@CsnKhai:/#
```

R (Running), S (Sleeping), D (Uninterruptible Sleep), Z (Zombie), T (Stopped), X (Dead), W (Paging), < (High-Priority), N (Low-Priority), < (Foreground), > (Background).

8. Display only the processes of a specific user.

```
root@CsnKhai:/# ps -u student
  PID TTY          TIME CMD
  850 tty1     00:00:00 bash
  884 ?        00:00:00 sshd
  885 pts/0    00:00:00 bash
  917 ?        00:00:00 sshd
  918 ?        00:00:00 sftp-server
```

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

```
root@CsnKhai:/# top
top - 20:01:33 up  2:15,  2 users,  load average: 0.00, 0.00, 0.00
Tasks:  68 total,   1 running,  67 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   119504 used,   128288 free,    13888 buffers
KiB Swap:        0 total,        0 used,        0 free.    73932 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   56 root      20   0       0      0      0 S  0.3  0.0   0:04.72 kworker/0:2
  884 student   20   0   11192   2592   1792 S  0.3  1.0   0:00.89 sshd
 1233 root      20   0    5420   1336    988 R  0.3  0.5   0:00.02 top
    1 root      20   0    4196   2180   1392 S  0.0  0.9   0:00.76 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.13 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.12 watchdog/0
   11 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 khelper
   12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kdevtmpfs
   13 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 netns
   14 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 writeback
   15 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kintegrityd
   16 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 bioset
   17 root       0 -20       0      0      0 S  0.0  0.0   0:00.03 kworker/u3:0
   18 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kblockd
   19 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 ata_sff
   20 root      20   0       0      0      0 S  0.0  0.0   0:00.26 khubd
   21 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 md
   22 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 devfreq_wq
   25 root      20   0       0      0      0 S  0.0  0.0   0:00.00 khungtaskd
   26 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kswapd0
   27 root      25   5       0      0      0 S  0.0  0.0   0:00.00 ksmd
   28 root      20   0       0      0      0 S  0.0  0.0   0:00.00 fsnotify_mark
   29 root      20   0       0      0      0 S  0.0  0.0   0:00.00 ecryptfs-kthrea
   30 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 crypto
   42 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kthrotld
   44 root      20   0       0      0      0 S  0.0  0.0   0:00.00 scsi_eh_0
```

10. What information does top command display?

PID: Process ID of each task.

USER: The owner of the process.

PR: Priority of the process.

NI: The nice value of the process.

VIRT: Virtual memory used by the process.

RES: Resident memory used by the process.

SHR: Shared memory used by the process.

S: Process status (running, sleeping, stopped, etc.).

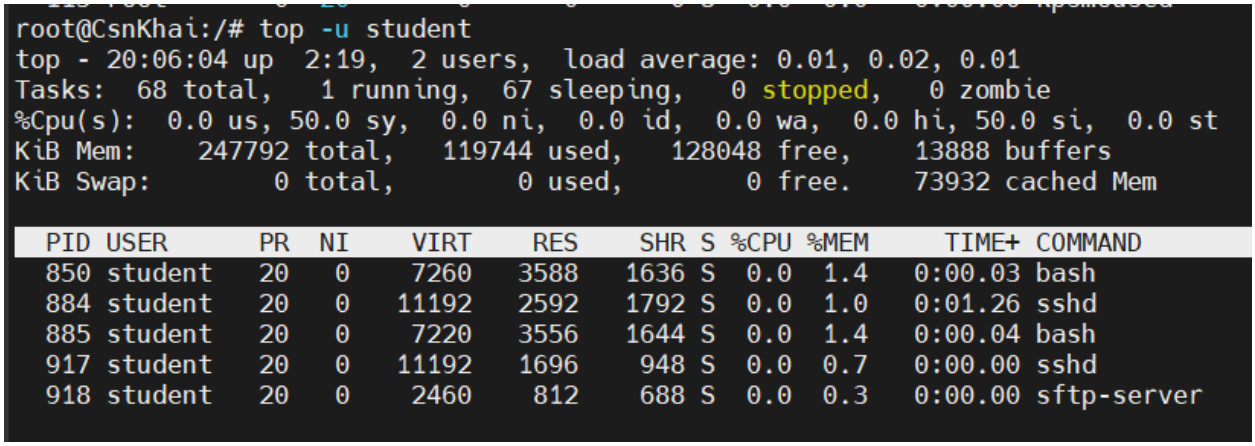
%CPU: Percentage of CPU utilization by the process.

%MEM: Percentage of physical memory (RAM) utilized by the process.

TIME+: Total accumulated CPU time used by the process.

COMMAND: The command that started the process.

12. Display the processes of the specific user using the top command.



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

- Sorting Processes:

F: Pressing the F key allows you to choose a sorting field. You'll be presented with a list of available fields (e.g., PID, %CPU, %MEM, etc.). Select a field by typing its corresponding key, and top will sort the processes based on that field.

- Changing Refresh Interval:

d or s: Pressing the d or s key allows you to change the update interval of top. You can enter a value in seconds to specify how often top should refresh its display. For example, typing 5 and pressing Enter would set the refresh interval to 5 seconds.

- Filtering Processes by User:

u: Pressing the u key prompts you to enter a username. After entering the username, top will only display processes owned by that user.

- Killing a Process:

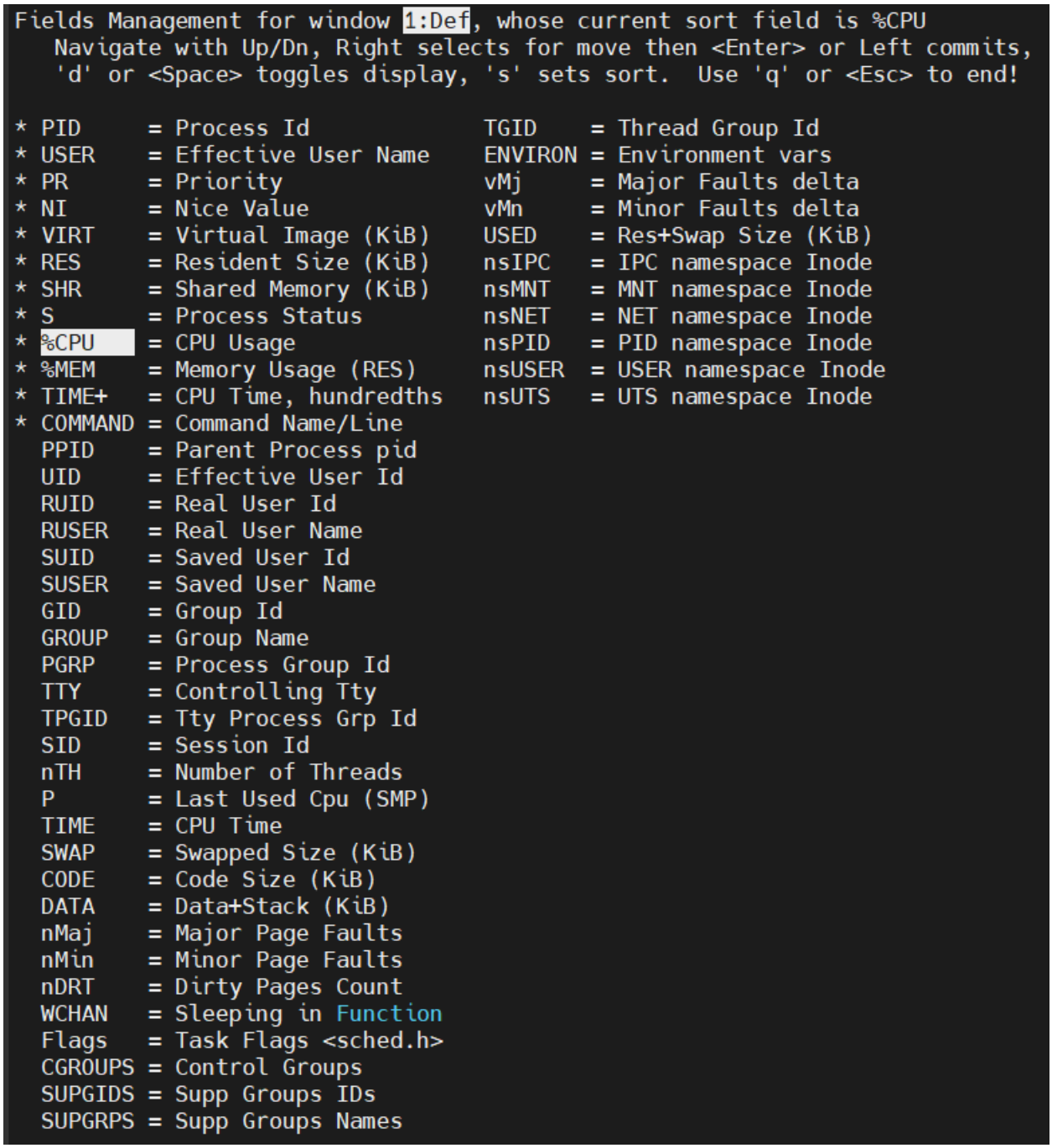
k: Pressing the k key allows you to enter a process ID (PID) of a process you want to terminate. After entering the PID, top will send a SIGTERM signal to the specified process, requesting it to gracefully terminate. You might need superuser privileges (root or sudo) to kill processes you don't own.

- Toggle Highlighting and Secure Mode:

H: Pressing the H key toggles highlighting of user and group names. When enabled, the highlighted usernames are the owners of the processes.

Z: Pressing the Z key toggles between normal and secure mode. In secure mode, top doesn't display the full command-line arguments of processes.

Press F:



Press u: user - student

```
top - 20:10:47 up  2:24,  2 users,  load average: 0.00, 0.01, 0.01
Tasks:  68 total,   3 running,  65 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   119864 used,   127928 free,    13888 buffers
KiB Swap:        0 total,        0 used,        0 free.    73932 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  884 student   20   0   11192   2592   1792 R  0.3  1.0   0:01.38 sshd
  850 student   20   0    7260   3588   1636 S  0.0  1.4   0:00.03 bash
  885 student   20   0    7220   3556   1644 S  0.0  1.4   0:00.04 bash
  917 student   20   0   11192   1696    948 S  0.0  0.7   0:00.00 sshd
  918 student   20   0    2460    812    688 S  0.0  0.3   0:00.00 sftp-server
```

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

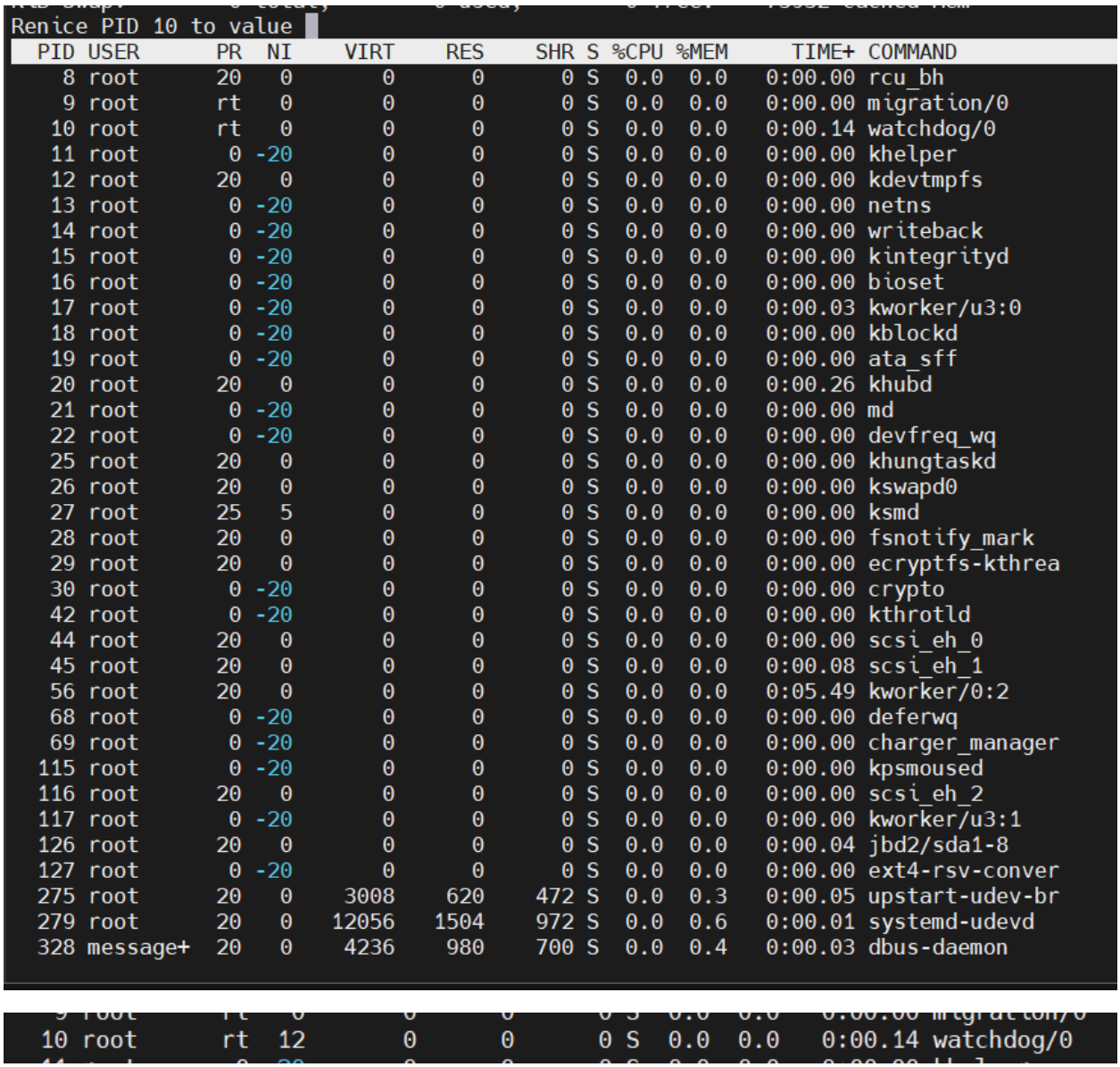
```
root@CsnKhai:/# top -o TIME+
top - 20:16:52 up  2:30,  2 users,  load average: 0.00, 0.01, 0.01
Tasks:  68 total,   1 running,  67 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    247792 total,   120296 used,   127496 free,    13896 buffers
KiB Swap:        0 total,        0 used,        0 free.    73932 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   56 root      20   0       0      0      0 S  0.0  0.0   0:05.33 kworker/0:2
  884 student   20   0   11192   2592   1792 S  0.0  1.0   0:01.89 sshd
    1 root      20   0    4196   2180   1392 S  0.0  0.9   0:00.76 init
 1214 root      20   0       0      0      0 S  0.0  0.0   0:00.48 kworker/u2:0
 1151 root      20   0       0      0      0 S  0.0  0.0   0:00.31 kworker/u2:2
   20 root      20   0       0      0      0 S  0.0  0.0   0:00.26 khubd
  923 root      20   0    5692   1940   1552 S  0.0  0.8   0:00.25 bash
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.15 rcu_sched
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.14 watchdog/0
   45 root      20   0       0      0      0 S  0.0  0.0   0:00.08 scsi_eh_1
  599 root      20   0    5512   2292    572 S  0.0  0.9   0:00.07 dhclient
  864 root      20   0   11192   3748   2988 S  0.0  1.5   0:00.07 sshd
  275 root      20   0    3008    620    472 S  0.0  0.3   0:00.05 upstart-udev-br
  126 root      20   0       0      0      0 S  0.0  0.0   0:00.04 jbd2/sda1-8
  885 student   20   0    7220   3556   1644 S  0.0  1.4   0:00.04 bash
   17 root       0 -20       0      0      0 S  0.0  0.0   0:00.03 kworker/u3:0
  328 message+  20   0    4236    980    700 S  0.0  0.4   0:00.03 dbus-daemon
  850 student   20   0    7260   3588   1636 S  0.0  1.4   0:00.03 bash
  866 root      20   0   11192   3732   2984 S  0.0  1.5   0:00.03 sshd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.02 ksoftirqd/0
  363 syslog    20   0   30476   1112    824 S  0.0  0.4   0:00.02 rsyslogd
  820 root      20   0    4400   2012   1532 S  0.0  0.8   0:00.02 login
  921 root      20   0    6740   2024   1604 S  0.0  0.8   0:00.02 sudo
  279 root      20   0   12056   1504    972 S  0.0  0.6   0:00.01 systemd-udevd
  386 root      20   0    3008    596    348 S  0.0  0.2   0:00.01 upstart-file-br
  466 root      20   0    2868    616    444 S  0.0  0.2   0:00.01 upstart-socket-
  758 root      20   0    3052    788    624 S  0.0  0.3   0:00.01 cron
 1240 root      20   0    5420   1348    992 R  0.0  0.5   0:00.01 top
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
```

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

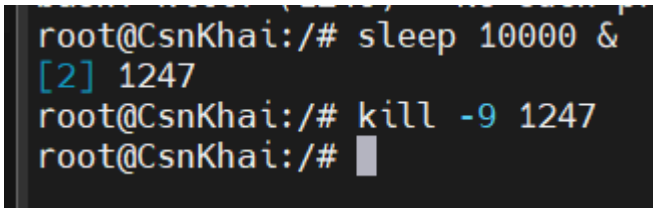
Commands: “nice”, “renice”.

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



While top is running, press the r -> print PID -> print priority.

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



- SIGTERM (15): Termination Signal

Purpose: This is the default signal sent by the kill command. It asks the process to terminate gracefully. The process can catch this signal and perform any cleanup operations before exiting.

Command: kill -15 PID or kill -TERM PID

- SIGKILL (9): Kill Signal

Purpose: This is a more forceful signal that immediately terminates the process without allowing it to perform any cleanup operations.

Command: kill -9 PID

- SIGINT (2): Interrupt Signal

Purpose: This signal is sent when the user presses Ctrl+C in the terminal. It's used to interrupt a process and request it to terminate.

Command: kill -2 PID

- SIGHUP (1): Hangup Signal

Purpose: Historically used to notify processes that the terminal has been disconnected. Now often used to request processes to reload their configuration.

Command: kill -1 PID

- SIGUSR1 (10) and SIGUSR2 (12): User-defined Signals

Purpose: These signals are user-defined and can be used for any custom purpose. They are not predefined by the system.

Command: kill -10 PID (SIGUSR1) or kill -12 PID (SIGUSR2)

- SIGSTOP (19): Stop Signal

Purpose: This signal suspends the process's execution. Unlike SIGKILL, it can be caught by the process, allowing for resumption later.

Command: kill -19 PID

- SIGCONT (18): Continue Signal

Purpose: This signal resumes the execution of a process that has been stopped with SIGSTOP or SIGTSTP.

Command: kill -18 PID

17. Commands jobs, fg, bg, nohup. What are they for? Use the sleep, yes command to

demonstrate the process control mechanism with fg, bg.

The commands jobs, fg, bg, and nohup are used for process control and management in Unix-like operating systems. They allow you to manage background and foreground processes, control job execution, and run processes that are immune to hangups. Let's look at each of these commands and their purposes:

- jobs Command:

Purpose: Displays a list of all the background and suspended processes associated with the current shell session.

Syntax: jobs

Example: Running multiple background tasks and then using jobs to list them.

- fg Command:

Purpose: Brings a background process to the foreground (active) so that it interacts with the user on the terminal.

Syntax: fg [job_spec]

Example: Using fg %1 to bring the first background job to the foreground.

- bg Command:

Purpose: Resumes a suspended or stopped background process in the background, allowing it to continue executing.

Syntax: bg [job_spec]

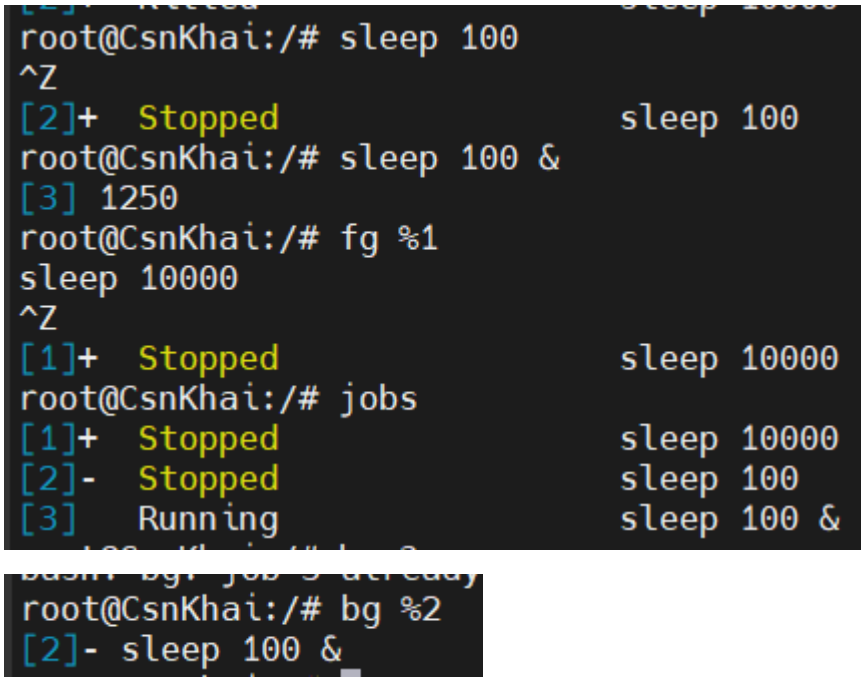
Example: Using bg %1 to resume the first stopped background job.

- nohup Command:

Purpose: Runs a command immune to hangups, ensuring that the command continues executing even after the terminal is closed.

Syntax: nohup command [arguments] &

Example: Running a long-lasting command using nohup.



Part2

1. Check the implementability of the most frequently used OPENSSH commands in the MS Windows operating system. (Description of the expected result of the commands + screenshots: command – result should be presented)

```
PS C:\Users\Yulia> ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Yulia/.ssh/id_rsa):
Created directory 'C:\Users\Yulia/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Yulia/.ssh/id_rsa.
Your public key has been saved in C:\Users\Yulia/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:VN3zLeTi2JoCFkVSBuytJpS233sM91RpIYiJZSmF+Oo yulia@DESKTOP-KJ5611F
The key's randomart image is:
+---[RSA 4096]----+
|     ooOB+.o .   |
|    . =+=.. o =  |
|     + +.    + =.|
|    + +..    . * o|
|   o o oS  + + . |
|    + = . o +    |
|   . = o + =     |
|    E . o = .    |
|        .+       |
+----[SHA256]-----+
PS C:\Users\Yulia>
```

```
PS C:\Users\Yulia> scp C:\Users\Yulia/.ssh/id_rsa.pub student@192.168.1.103:/home/student/.ssh/yulia_id_rsa.pub
student@192.168.1.103's password:
id_rsa.pub                                    100%  748   374.4KB/s   00:00
PS C:\Users\Yulia>
```

```
PS C:\Users\Yulia> ssh -i C:\Users\Yulia/.ssh/id_rsa.pub student@192.168.1.103
student@192.168.1.103's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-63-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Aug 17 17:50:47 2023 from 192.168.1.104
student@CsnKhai:~$
```

2. Implement basic SSH settings to increase the security of the client-server connection (at least

```
root@CsnKhai:/# vim /etc/ssh/sshd_config
```

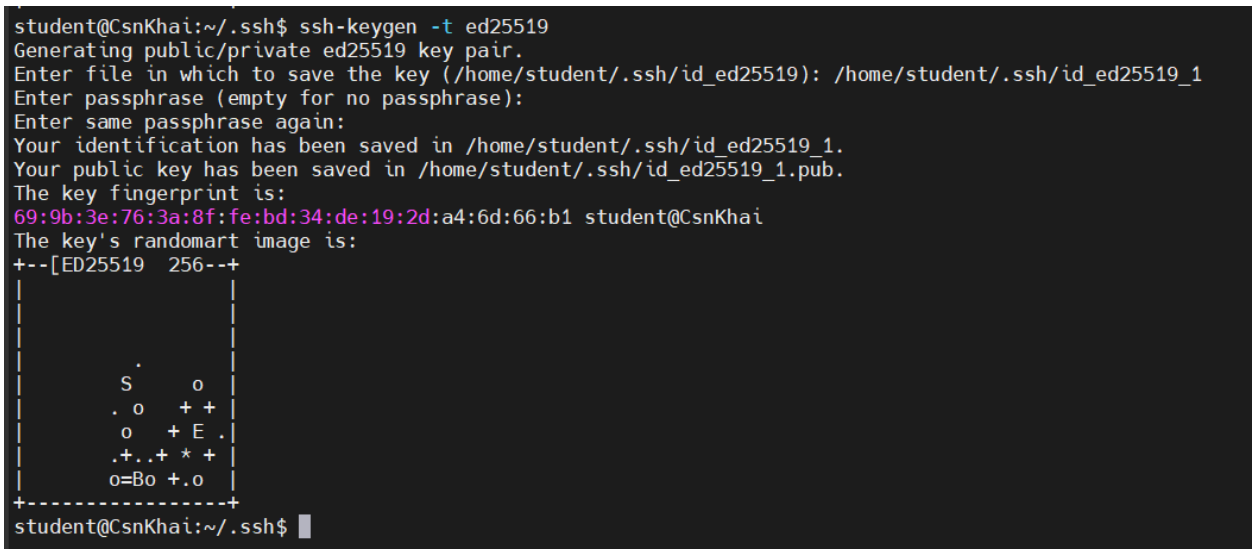
```
# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
```

```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no
```

3. List the options for choosing keys for encryption in SSH. Implement 3 of them.

```
student@CsnKhai:~/.ssh$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): /home/student/.ssh/id_rsa_1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/id_rsa_1.
Your public key has been saved in /home/student/.ssh/id_rsa_1.pub.
The key fingerprint is:
a8:80:72:de:5c:e6:41:23:32:f9:8e:a9:43:8d:12:a6 student@CsnKhai
The key's randomart image is:
+--[ RSA 4096]----+
|                 |
|    .            |
|  + . o          |
|.o + o o         |
|=.= . = S        |
|E= O = .         |
|o + = .          |
|..               |
|..               |
+-----------------+
student@CsnKhai:~/.ssh$
```

```
Enter file in which to save the key (/home/student/.ssh/id_ecdsa): /home/student/.ssh/id_ecdsa_1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/id_ecdsa_1.
Your public key has been saved in /home/student/.ssh/id_ecdsa_1.pub.
The key fingerprint is:
a3:c0:64:ea:09:8a:0b:da:d6:02:84:af:fe:85:91:5d student@CsnKhai
The key's randomart image is:
+--[ECDSA  256]---+
|                 |
|                 |
|.   o  E         |
|.. =o .          |
|o..oo.  S        |
|+o..o. . .       |
|+ooo ..          |
|+oo o            |
|+ooo             |
+-----------------+
student@CsnKhai:~/.ssh$
```



- AES (Advanced Encryption Standard):

AES is a widely used symmetric encryption algorithm. It offers strong encryption and efficient performance. SSH supports different key sizes for AES, such as 128, 192, and 256 bits. AES encryption provides confidentiality for data transmitted between the client and server.

To implement AES encryption in SSH, you don't need to explicitly choose it as an option. SSH implementations automatically negotiate the encryption algorithms based on their capabilities and preferences.

- RSA (Rivest-Shamir-Adleman):

RSA is an asymmetric encryption algorithm used for key exchange and digital signatures. In SSH, RSA is often used for key-based authentication. The client generates a key pair (public and private), and the public key is sent to the server to authenticate the client.

- ECDSA (Elliptic Curve Digital Signature Algorithm):

ECDSA is another asymmetric encryption algorithm that offers strong security with shorter key lengths compared to RSA. It's commonly used for digital signatures and key exchange in SSH.

- ChaCha20-Poly1305:

ChaCha20 is a stream cipher, and Poly1305 is an authenticator. Together, they provide a strong and efficient option for encryption and message authentication. This combination is often used as an alternative to AES-based ciphers.

To use ChaCha20-Poly1305 in SSH, you typically don't need to explicitly choose it. Many modern SSH implementations support it as part of their cipher suites.

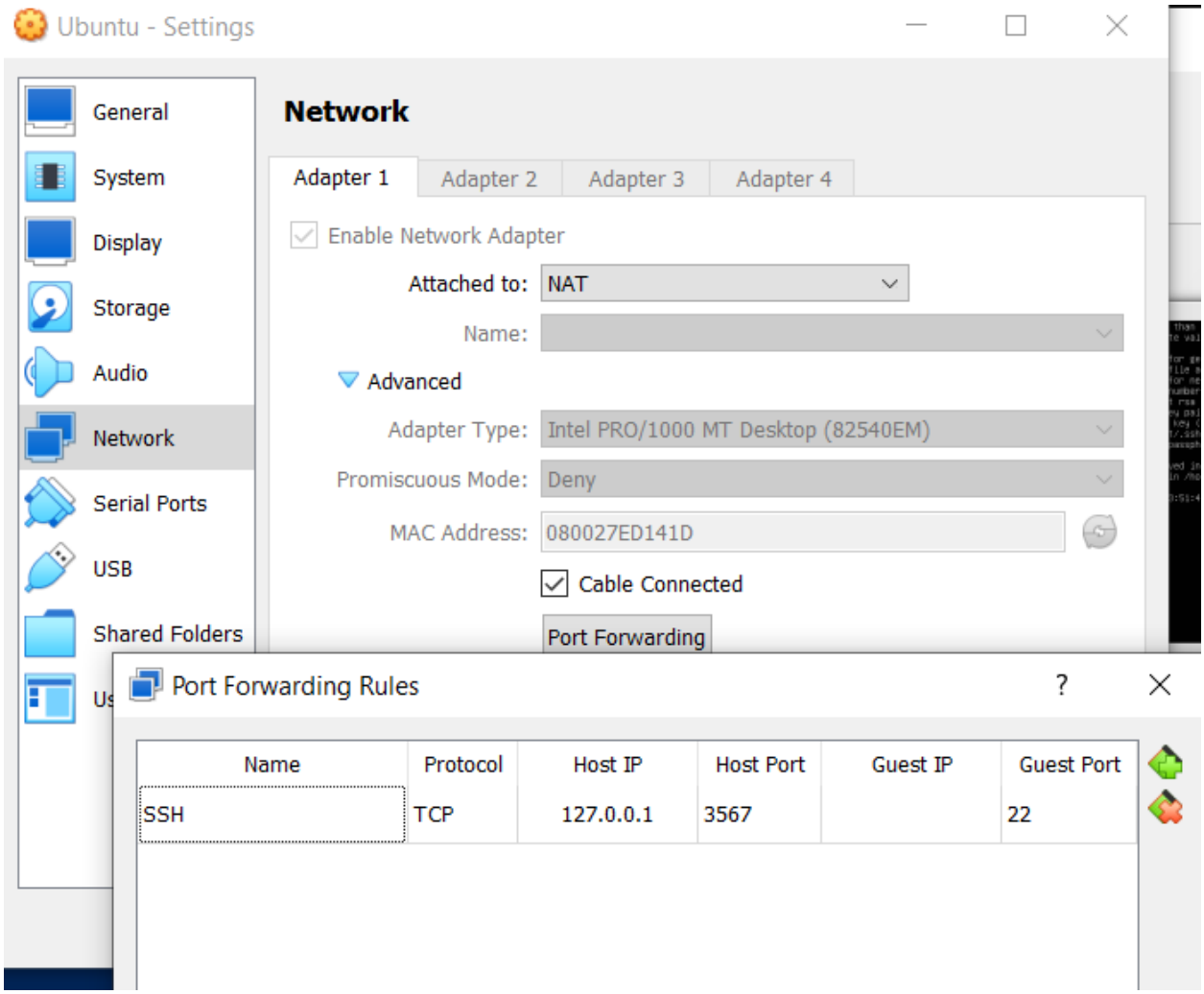
- Ed25519:

Ed25519 is a modern elliptic curve digital signature algorithm that offers strong security with relatively small key sizes. It's becoming increasingly popular for key-based authentication in SSH.

- Diffie-Hellman Group Exchange:

Diffie-Hellman (DH) is used for key exchange in SSH. Different DH groups offer varying levels of security.

4. Implement port forwarding for the SSH client from the host machine to the guest Linux virtual machine behind NAT.




```
PS C:\Users\Yulia> ssh -p 3567 -i C:\Users\Yulia/.ssh/id_rsa student@localhost
student@localhost's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-63-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Aug 17 20:37:30 2023 from 10.0.2.2
student@CsnKhai:~$
```

5*. Intercept (capture) traffic (tcpdump, wireshark) while authorizing the remote client on the server using ssh, telnet, rlogin. Analyze the result.

```
20:52:15.124699 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3344:3488, ack 97, win 40880, length 144
20:52:15.125602 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3488, win 65535, length 0
20:52:16.127323 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3488:3584, ack 97, win 40880, length 96
20:52:16.128235 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3584, win 65535, length 0
20:52:16.447263 IP 10.0.2.2.60194 > 10.0.2.15.ssh: Flags [R.], seq 166466223, ack 4037387730, win 65535, length 0
20:52:17.128633 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3584:3664, ack 97, win 40880, length 80
20:52:17.129567 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3664, win 65535, length 0
20:52:17.129706 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3664:3760, ack 97, win 40880, length 96
20:52:17.130487 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3760, win 65535, length 0
20:52:18.130060 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3760:3872, ack 97, win 40880, length 112
20:52:18.130988 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3872, win 65535, length 0
20:52:19.131310 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3872:3952, ack 97, win 40880, length 80
20:52:19.132162 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 3952, win 65535, length 0
20:52:20.132641 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 3952:4032, ack 97, win 40880, length 80
20:52:20.133492 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 4032, win 65535, length 0
20:52:20.891042 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [S], seq 179648001, win 65535, options [mss 1460], length 0
20:52:20.891110 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [S.], seq 3671857462, ack 179648002, win 29200, options [mss 1460], length 0
20:52:20.891510 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 1, win 65535, length 0
20:52:20.895725 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1:34, ack 1, win 65535, length 33
20:52:20.895751 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [.], ack 34, win 29200, length 0
20:52:20.901887 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 1:44, ack 34, win 29200, length 43
20:52:20.902185 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 44, win 65535, length 0
20:52:20.902556 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [.], seq 44:1504, ack 34, win 29200, length 1460
20:52:20.902574 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 1504:1692, ack 34, win 29200, length 188
20:52:20.902902 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 1504, win 65535, length 0
20:52:20.902913 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 1692, win 65535, length 0
20:52:20.902915 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 34:1426, ack 1692, win 65535, length 1392
20:52:20.905090 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1426:1474, ack 1692, win 65535, length 48
20:52:20.905098 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [.], ack 1474, win 32016, length 0
20:52:20.908851 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 1692:1972, ack 1474, win 32016, length 280
20:52:20.909126 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 1972, win 65535, length 0
20:52:20.915579 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1474:1490, ack 1972, win 65535, length 16
20:52:20.915590 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1490:1534, ack 1972, win 65535, length 44
20:52:20.915626 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [.], ack 1534, win 32016, length 0
20:52:20.915661 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 1972:2016, ack 1534, win 32016, length 44
20:52:20.915925 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 2016, win 65535, length 0
20:52:20.916014 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1534:1602, ack 2016, win 65535, length 68
20:52:20.925377 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 2016:2068, ack 1602, win 32016, length 52
20:52:20.925824 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 2068, win 65535, length 0
20:52:20.925994 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 1602:2222, ack 2068, win 65535, length 620
20:52:20.926392 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 2068:2120, ack 2222, win 34800, length 52
```

```
20:52:31.151843 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 5760, win 65535, length 0
20:52:31.840313 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 2618:2654, ack 2580, win 65535, length 36
20:52:31.843098 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 2580:2624, ack 2654, win 40368, length 44
20:52:31.844873 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 2624, win 65535, length 0
20:52:31.845630 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 2624:2660, ack 2654, win 40368, length 36
20:52:31.846433 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 2660, win 65535, length 0
20:52:31.846608 IP 10.0.2.15.ssh > 10.0.2.2.60196: Flags [P.], seq 2660:2800, ack 2654, win 40368, length 140
20:52:31.846905 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [.], ack 2800, win 65535, length 0
20:52:31.847554 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [P.], seq 2654:2750, ack 2800, win 65535, length 96
20:52:31.848333 IP 10.0.2.2.60196 > 10.0.2.15.ssh: Flags [R.], seq 2750, ack 2800, win 65535, length 0
20:52:32.152720 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 5760:6016, ack 193, win 40880, length 256
20:52:32.153660 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6016, win 65535, length 0
20:52:33.153037 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6016:6096, ac
k 193, win 40880, length 80
20:52:33.154065 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6096, win 6553
5, length 0
20:52:33.991868 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [P.], seq 193:273, ack
6096, win 65535, length 80
20:52:34.031583 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [.], ack 273, win 40880
, length 0
20:52:34.154439 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6096:6208, ac
k 273, win 40880, length 112
20:52:34.155095 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6208, win 6553
5, length 0
20:52:35.156012 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6208:6304, ack 273, win 40880, length 96
20:52:35.157086 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6304, win 65535, length 0
20:52:35.615965 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [P.], seq 273:337, ack 6304, win 65535, length 64
20:52:35.616005 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [.], ack 337, win 40880, length 0
20:52:36.157326 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6304:6432, ack 337, win 40880, length 128
20:52:36.157928 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6432, win 65535, length 0
20:52:37.158654 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6432:6512, ack 337, win 40880, length 80
20:52:37.159321 IP 10.0.2.2.60156 > 10.0.2.15.ssh: Flags [.], ack 6512, win 65535, length 0
20:52:38.160715 IP 10.0.2.15.ssh > 10.0.2.2.60156: Flags [P.], seq 6512:6592, ack 337, win 40880, length 80
```

On these screens we can see how we login and logout using SSH.