

TASK5.3

Part1

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

one of five Linux process states: running & runnable, interruptable_sleep, uninterruptable_sleep, stopped, and zombie

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

```
Ubuntu 14.04.3 LTS vm1 tty1

vm1 login: student
Password:
Last login: Fri Feb 18 09:52:27 UTC 2022 on tty1
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-63-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@vm1:~$ pstree -h
init--cron
      |--dbus-daemon
      |--dhclient
      |--dnsmasq
      |--5*[getty]
      |--login--bash--pstree
      |--ondemand--sleep
      |--rsyslogd--3*[{rsyslogd}]
      |--sshd
      |--systemd-logind
      |--systemd-udev--systemd-udev
      |--upstart-file-br
      |--upstart-socket-
      |--upstart-udev-br
student@vm1:~$
```

3. What is a proc file system?

Proc file system (procfs) is virtual file system created on fly when system boots and is dissolved at time of system shut down. It contains useful information about the processes that are currently running, it is regarded as control and information center for kernel.

ls -l /proc | less

```
total 0
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 1
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 10
dr-xr-xr-x  9 root      root           0 Feb 18 11:27 1020
dr-xr-xr-x  9 student    student        0 Feb 18 11:28 1042
dr-xr-xr-x  9 student    student        0 Feb 18 11:28 1043
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 11
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 114
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 115
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 116
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 117
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 12
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 128
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 129
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 13
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 14
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 15
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 16
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 17
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 18
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 19
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 2
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 20
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 21
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 22
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 23
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 25
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 258
dr-xr-xr-x  9 root      root           0 Feb 18 11:10 26
:
```

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

`less /proc/cpuinfo`

```
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 158
model name     : Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
stepping       : 10
cpu MHz        : 2403.661
cache size     : 8192 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    : 22
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 movbe popcnt aes xsave a
vx rdrand lahf_lm abm 3dnowprefetch fsgsbase avx2 invpcid rdseed
bogomips       : 4807.32
clflush size   : 64
cache_alignme  : 64
address sizes   : 39 bits physical, 48 bits virtual
:.
```

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.

`ps o user,args,group | less`

```
root      [kpsmoused]                root
root      [kworker/0:2]              root
root      [kworker/u3:1]             root
root      [scsi_eh_2]                root
root      [jbd2/sda1-8]              root
root      [ext4-rsv-conver]          root
root      upstart-udev-bridge --daemo root
root      /lib/systemd/systemd-udev root
root      dhclient -1 -v -pf /run/dhc root
message+  dbus-daemon --system --fork messagebus
syslog    rsyslogd                   syslog
root      /lib/systemd/systemd-logind root
root      upstart-socket-bridge --dae root
root      upstart-file-bridge --daemo root
root      /sbin/getty -8 38400 tty4   root
root      /sbin/getty -8 38400 tty5   root
root      /sbin/getty -8 38400 tty2   root
root      /sbin/getty -8 38400 tty3   root
root      /sbin/getty -8 38400 tty6   root
root      /usr/sbin/sshd -D           root
root      cron                       root
dnsmasq   /usr/sbin/dnsmasq -x /var/r dip
root      /bin/login --               student
root      [kauditd]                  root
student   -bash                      student
root      [kworker/u2:2]              root
root      [kworker/u2:0]              root
student   ps -eo user,args,group      student
student   less                       student
(END)
```

6. How to define kernel processes and user processes?

ps aux| less - kernel's processes have [brackets] in command column, user's not

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?

```
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.8  4184  2212 ?        Ss   11:10   0:00 /sbin/init
root         2  0.0  0.0      0      0 ?        S    11:10   0:00 [kthreadd]
root         3  0.0  0.0      0      0 ?        S    11:10   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0      0 ?        S<   11:10   0:00 [kworker/0:0H]
root         7  0.0  0.0      0      0 ?        S    11:10   0:00 [rcu_sched]
root         8  0.0  0.0      0      0 ?        S    11:10   0:00 [rcu_bh]
root         9  0.0  0.0      0      0 ?        S    11:10   0:00 [migration/0]
root        10  0.0  0.0      0      0 ?        S    11:10   0:00 [watchdog/0]
root        11  0.0  0.0      0      0 ?        S<   11:10   0:00 [khelper]
root        12  0.0  0.0      0      0 ?        S    11:10   0:00 [kdevtmpfs]
root        13  0.0  0.0      0      0 ?        S<   11:10   0:00 [netns]
root        14  0.0  0.0      0      0 ?        S<   11:10   0:00 [writeback]
root        15  0.0  0.0      0      0 ?        S<   11:10   0:00 [kintegrityd]
root        16  0.0  0.0      0      0 ?        S<   11:10   0:00 [bioset]
root        17  0.0  0.0      0      0 ?        S<   11:10   0:00 [kworker/u3:0]
root        18  0.0  0.0      0      0 ?        S<   11:10   0:00 [kblockd]
root        19  0.0  0.0      0      0 ?        S<   11:10   0:00 [ata_sff]
root        20  0.0  0.0      0      0 ?        S    11:10   0:00 [khubd]
root        21  0.0  0.0      0      0 ?        S<   11:10   0:00 [md]
root        22  0.0  0.0      0      0 ?        S<   11:10   0:00 [devfreq_wq]
root        23  0.0  0.0      0      0 ?        S    11:10   0:03 [kworker/0:1]
root        25  0.0  0.0      0      0 ?        S    11:10   0:00 [khungtaskd]
root        26  0.0  0.0      0      0 ?        S    11:10   0:00 [kswapd0]
root        27  0.0  0.0      0      0 ?        SN   11:10   0:00 [ksmd]
root        28  0.0  0.0      0      0 ?        S    11:10   0:00 [fsnotify_mark]
root        29  0.0  0.0      0      0 ?        S    11:10   0:00 [ecryptfs-kthre
a]
root        30  0.0  0.0      0      0 ?        S<   11:10   0:00 [crypto]
:~
```

PROCESS STATE CODES

Here are the different values that the s, stat and state output specifiers (header "STAT" or "S") will display to describe the state of a process:

D	uninterruptible sleep (usually IO)
R	running or runnable (on run queue)
S	interruptible sleep (waiting for an event to complete)
T	stopped, either by a job control signal or because it is being traced
W	paging (not valid since the 2.6.xx kernel)
X	dead (should never be seen)
Z	defunct ("zombie") process, terminated but not reaped by its parent

For BSD formats and when the stat keyword is used, additional characters may be displayed:

<	high-priority (not nice to other users)
N	low-priority (nice to other users)
L	has pages locked into memory (for real-time and custom IO)
s	is a session leader
l	is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+	is in the foreground process group

OBSOLETE SORT KEYS

These keys are used by the BSD O option (when it is used for sorting).

Manual page ps(1) line 514 (press h for help or q to quit)

8. Display only the processes of a specific user.

```
student@vm1:~$ ps u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   959  0.0  1.2  6668  3116 tty1    S      11:10   0:00 -bash
student  1231  0.0  0.4  5216  1152 tty1    R+     12:22   0:00 ps u
student@vm1:~$ ps
  PID TTY          TIME CMD
  959 tty1      00:00:00 bash
 1232 tty1      00:00:00 ps
student@vm1:~$ _
```

ps -U root (or another specific username)

```
 42 ?          00:00:00 kthrotld
 44 ?          00:00:00 scsi_eh_0
 45 ?          00:00:00 scsi_eh_1
 67 ?          00:00:00 deferwq
 68 ?          00:00:00 charger_manager
114 ?          00:00:00 kpsmouse
115 ?          00:00:00 kworker/0:2
116 ?          00:00:00 kworker/u3:1
117 ?          00:00:00 scsi_eh_2
128 ?          00:00:00 jbd2/sda1-8
129 ?          00:00:00 ext4-rsv-conver
258 ?          00:00:00 upstart-udev-br
264 ?          00:00:00 systemd-udev
555 ?          00:00:00 dhclient
647 ?          00:00:00 systemd-logind
670 ?          00:00:00 upstart-socket-
679 ?          00:00:00 upstart-file-br
706 tty4       00:00:00 getty
708 tty5       00:00:00 getty
711 tty2       00:00:00 getty
712 tty3       00:00:00 getty
714 tty6       00:00:00 getty
738 ?          00:00:00 sshd
788 ?          00:00:00 cron
890 tty1       00:00:00 login
941 ?          00:00:00 kauditd
1205 ?         00:00:00 kworker/u2:2
1206 ?         00:00:00 kworker/u2:0
1236 ?         00:00:00 kworker/u2:1
student@vm1:~$ ps -U root_
```

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

using different flags with ps

```
Usage:
ps [options]

Basic options:
-A, -e          all processes
-a            all with tty, except session leaders
-a            all with tty, including other users
-d            all except session leaders
-N, --deselect  negate selection
-r            only running processes
-T            all processes on this terminal
-x            processes without controlling ttys

Selection by list:
-C <command>    command name
-G, --Group <gid>  real group id or name
-g, --group <group> session or effective group name
-p, --pid <pid>    process id
--ppid <pid>      select by parent process id
-s, --sid <session> session id
-t, t, --tty <tty> terminal
-u, U, --user <uid> effective user id or name
-U, --User <uid>   real user id or name

selection <arguments> take either:
comma-separated list e.g. '-u root,nobody' or
blank-separated list e.g. '-p 123 4567'

:

--headers      repeat header lines, one per page
--no-headers    do not print header at all
--cols, --columns, --width <num>
               set screen width
--rows, --lines <num>
               set screen height

Show threads:
-H            as if they were processes
-L            possibly with LWP and NLWP columns
-m, m        after processes
-T            possibly with SPID column

Miscellaneous options:
-c            show scheduling class with -l option
-c            show true command name
-e            show the environment after command
-k, --sort    specify sort order as: [+|-]key[, [+|-]key[,...]]
-L            list format specifiers
-n            display numeric uid and wchan
-S, --cumulative include some dead child process data
-y            do not show flags, show rss (only with -l)
-V, V, --version display version information and exit
-w, w        unlimited output width

--help <simple|list|output|threads|misc|all>
               display help and exit

For more details see ps(1).
(END)
```

10. What information does top command display?

The `top` program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts

```
top - 12:40:25 up 1:29, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 61 total, 1 running, 60 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 94.0 id, 6.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 247792 total, 75132 used, 172660 free, 14080 buffers
KiB Swap: 0 total, 0 used, 0 free. 36816 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1285	student	20	0	5420	1308	988	R	0.3	0.5	0:00.02	top
1	root	20	0	4184	2212	1392	S	0.0	0.9	0:00.92	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
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	0:00.44	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	0:00.07	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	0:00.00	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	0:00.00	khubb
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
23	root	20	0	0	0	0	S	0.0	0.0	0:04.71	kworker/0:1
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd

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

```
top - 12:44:04 up 1:33, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 60 total, 1 running, 59 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, 75112 used, 172680 free, 14080 buffers
KiB Swap: 0 total, 0 used, 0 free. 36848 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	4184	2212	1392	S	0.0	0.9	0:00.92	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
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	0:00.45	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	0:00.08	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	0:00.00	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	0:00.00	khubb
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
23	root	20	0	0	0	0	S	0.0	0.0	0:04.90	kworker/0:1
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

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

shift+m – sort by memory usage

c – shows absolute path of the command

```
top - 12:46:16 up 1:35, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 60 total, 1 running, 59 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, 75160 used, 172632 free, 14080 buffers
KiB Swap: 0 total, 0 used, 0 free. 36848 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
738	root	20	0	7796	2484	1988	S	0.0	1.0	0:00.00	/usr/sbin/s+
1	root	20	0	4184	2212	1392	S	0.0	0.9	0:00.92	/sbin/init
890	root	20	0	4400	2008	1532	S	0.0	0.8	0:00.03	/bin/login +
555	root	20	0	5512	1860	140	S	0.0	0.8	0:00.00	dhclient -i+
647	root	20	0	4212	1732	1440	S	0.0	0.7	0:00.00	/lib/system+
264	root	20	0	12052	1504	972	S	0.0	0.6	0:00.26	/lib/system+
670	root	20	0	3132	876	452	S	0.0	0.4	0:00.03	upstart-soc+
706	root	20	0	4644	840	716	S	0.0	0.3	0:00.00	/sbin/getty+
708	root	20	0	4644	836	716	S	0.0	0.3	0:00.00	/sbin/getty+
714	root	20	0	4644	836	716	S	0.0	0.3	0:00.00	/sbin/getty+
712	root	20	0	4644	832	716	S	0.0	0.3	0:00.00	/sbin/getty+
711	root	20	0	4644	824	716	S	0.0	0.3	0:00.00	/sbin/getty+
788	root	20	0	3052	788	624	S	0.0	0.3	0:00.00	cron
258	root	20	0	3008	616	472	S	0.0	0.2	0:00.18	upstart-ude+
679	root	20	0	3012	616	344	S	0.0	0.2	0:00.04	upstart-fil+
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/+
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	[kworker/0:+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.45	[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/+
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	[watchdog/0]
11	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	[khelper]

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

```
top - 12:48:38 up 1:37, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 60 total, 1 running, 59 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, 75116 used, 172676 free, 14080 buffers
KiB Swap: 0 total, 0 used, 0 free. 36848 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
23	root	20	0	0	0	0	S	0.0	0.0	0:05.14	[kworker/0:+
1	root	20	0	4184	2212	1392	S	0.0	0.9	0:00.94	/sbin/init
7	root	20	0	0	0	0	S	0.0	0.0	0:00.46	[rcu_sched]
1250	root	20	0	0	0	0	S	0.0	0.0	0:00.30	[kworker/u2+
264	root	20	0	12052	1504	972	S	0.0	0.6	0:00.28	/lib/system+
1281	root	20	0	0	0	0	S	0.0	0.0	0:00.26	[kworker/u2+
258	root	20	0	3008	616	472	S	0.0	0.2	0:00.19	upstart-ude+
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.08	[watchdog/0]
670	root	20	0	3132	876	452	S	0.0	0.4	0:00.04	upstart-soc+
679	root	20	0	3012	616	344	S	0.0	0.2	0:00.04	upstart-fil+
890	root	20	0	4400	2008	1532	S	0.0	0.8	0:00.03	/bin/login +
788	root	20	0	3052	788	624	S	0.0	0.3	0:00.01	cron
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/+
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	[kworker/0:+
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/+
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	[kintegrity+
16	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	[bioset]

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

When you only have one or a limited number of CPUs, you need to decide how to share those limited CPU resources among several competing processes. This is generally done by selecting one process for execution and letting it run for a short period (called a timeslice), or until it needs to wait for some event, such as IO to complete. To ensure that important processes don't get starved out by CPU hogs, the selection is done based on a scheduling priority.

nice/renice commands

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

Top → press 'r' → enter pid of process → enter renice value

```
top - 12:59:25 up 1:48, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 61 total, 1 running, 60 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.8 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 247792 total, 74852 used, 172940 free, 14080 buffers
KiB Swap: 0 total, 0 used, 0 free, 36848 cached Mem
PID to renice [default pid = 1]
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	4184	2212	1392	S	0.0	0.9	0:00.95	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
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	0:00.47	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	0:00.09	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	bioaset
17	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	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	0:00.00	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
23	root	20	0	0	0	0	S	0.0	0.0	0:05.68	kworker/0:1
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

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

```
root@vm1:/home/student# dd if=/dev/zero of=/dev/null &
[1] 1919
root@vm1:/home/student# dd if=/dev/zero of=/dev/null &
[2] 1920
root@vm1:/home/student# dd if=/dev/zero of=/dev/null &
[3] 1921
root@vm1:/home/student# ps fax | grep -B5 dd
  PID TTY          STAT       TIME COMMAND
    2 ?            S          0:00 [kthreadd]
--
 1836 tty1        Ss         0:00 /bin/login --
 1858 tty1        S          0:00 \_ -bash
 1899 tty1        S          0:00 \_ sudo su
 1900 tty1        S          0:00 \_ su
 1901 tty1        S          0:00 \_ bash
 1919 tty1        R          0:02 \_ dd if=/dev/zero of=/dev/null
 1920 tty1        R          0:01 \_ dd if=/dev/zero of=/dev/null
 1921 tty1        R          0:00 \_ dd if=/dev/zero of=/dev/null
 1922 tty1        R+         0:00 \_ ps fax
 1923 tty1        R+         0:00 \_ grep --color=auto -B5 dd
root@vm1:/home/student# kill -9 1858_
```


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 – see the list of tasks started in command line

fg – return to running task

bg – return to stopped task

```
root@vm1:/home/student# fg
dd if=/dev/zero of=/dev/null
^Z
[3]+  Stopped                  dd if=/dev/zero of=/dev/null
root@vm1:/home/student# ls -l | grep nohup
root@vm1:/home/student# nohup dd if=/dev/zero of=/dev/null &
[4] 2023
root@vm1:/home/student# nohup: ignoring input and appending output to 'nohup.out'
kill 2023
root@vm1:/home/student# ls -l | grep nohup
-rw----- 1 root root 0 Feb 18 15:01 nohup.out
[4]-  Terminated              nohup dd if=/dev/zero of=/dev/null
root@vm1:/home/student# kill -9 2023
bash: kill: (2023) - No such process
root@vm1:/home/student# ps fax | grep -B5 dd
  PID TTY          STAT TIME  COMMAND
  --  --
 1926 tty1      Ss    0:00  /bin/login --
 1949 tty1      S      0:00  \_ -bash
 1980 tty1      S      0:00  \_ sudo su
 1982 tty1      S      0:00  \_ su
 1983 tty1      S      0:00  \_ bash
 2002 tty1      R      2:07  \_ dd if=/dev/zero of=/dev/null
 2003 tty1      R      2:06  \_ dd if=/dev/zero of=/dev/null
 2004 tty1      T      0:05  \_ dd if=/dev/zero of=/dev/null
 2029 tty1      R+     0:00  \_ ps fax
 2030 tty1      R+     0:00  \_ grep --color=auto -B5 dd
root@vm1:/home/student#
```

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)

ssh – OpenSSH client (remote login program)

ssh-keygen – authentication key generation, managemeAuthentication key generation, management, and conversionnt, and conversion

```
yaroslav@nitro-5:~$ ssh -p 2223 student@192.168.0.108
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-63-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Feb 18 19:47:41 2022 from 10.0.2.2
student@vm1:~$
```

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

PermitRootLogin no

AllowUsers student

```
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
AllowUsers student

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys
:
```

PasswordAuthentication no

```
GNU nano 2.2.6      File: /etc/ssh/sshd_config      Modified
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

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

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

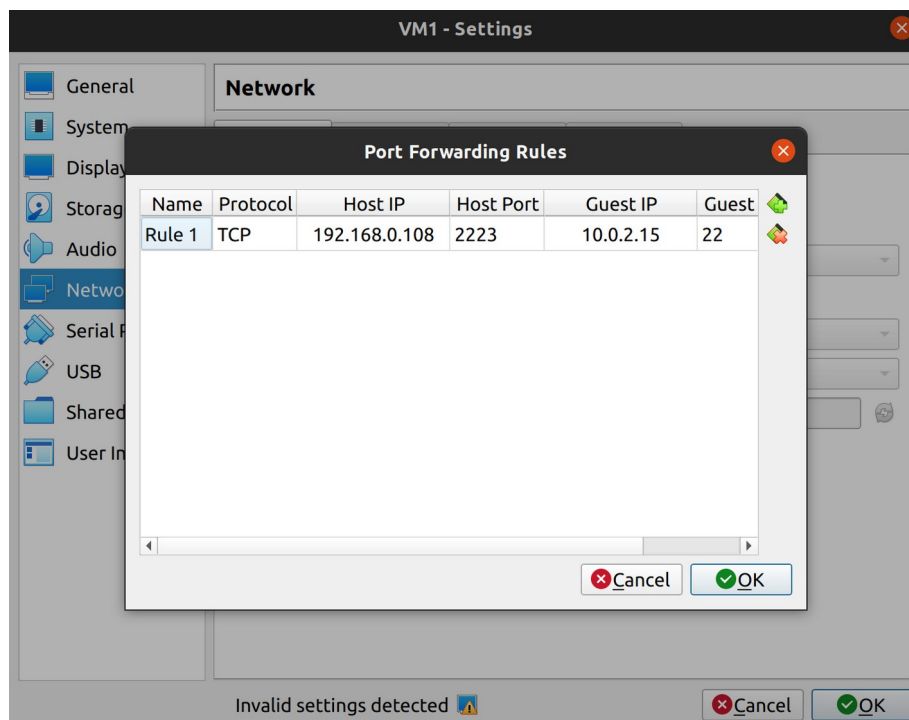
# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

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

```
student@vm1:~$ ssh-keygen -t ecdsa -b 384
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/id_ecdsa.
Your public key has been saved in /home/student/.ssh/id_ecdsa.pub.
The key fingerprint is:
63:0e:1f:a3:c7:d4:34:c6:a0:b0:51:6f:6c:11:89:db student@vm1
The key's randomart image is:
+--[ECDSA 384]--+
|  o...+o      |
|  +..+.      |
|  .o= =      |
|  .oE+ .      |
|  .S .      |
|  O +      |
|  . =      |
|  .      |
+-----+
student@vm1:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/student/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

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



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

```
15:53:18.815145 IP 10.0.2.15.10324 > 10.0.2.3.domain: 62458+ PTR? 3.2.0.10.in-ad
dr.arpa. (39)
15:53:18.873155 IP 10.0.2.3.domain > 10.0.2.15.10324: 62458 NXDomain 0/0/0 (39)
15:53:21.313993 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2222:2370, ac
k 2120, win 65535, length 148
15:53:21.332052 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2120:2148, ac
k 2370, win 37960, length 28
15:53:21.332638 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2148:2192, ac
k 2482, win 37960, length 44
15:53:21.332893 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2192:2300, ac
k 2300, win 37960, length 108
15:53:21.369214 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2300:2576, ac
k 3578, win 40880, length 276
15:53:21.369326 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2576:2644, ac
k 3578, win 40880, length 68
15:53:21.370262 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2644:2653, ac
k 3578, win 40880, length 9
15:53:21.370414 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
15:53:21.370434 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
15:53:21.404075 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2576:2644, ac
k 3578, win 40880, length 68
15:53:21.404511 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2644:2653, ac
k 3578, win 40880, length 9
15:53:21.404511 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
```

```
k 2120, win 65535, length 148
15:53:21.332052 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2120:2148, ac
k 2370, win 37960, length 28
15:53:21.332638 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2148:2192, ac
k 2482, win 37960, length 44
15:53:21.332893 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2192:2300, ac
k 2300, win 37960, length 108
15:53:21.369214 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2300:2576, ac
k 3578, win 40880, length 276
15:53:21.369326 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2576:2644, ac
k 3578, win 40880, length 68
15:53:21.370262 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2644:2653, ac
k 3578, win 40880, length 9
15:53:21.370414 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
15:53:21.370434 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
15:53:21.404075 IP 10.0.2.15.ssh > 10.0.2.2.33002: Flags [P.], seq 2576:2644, ac
k 3578, win 40880, length 68
15:53:21.404511 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2644:2653, ac
k 3578, win 40880, length 9
15:53:21.404511 IP 10.0.2.2.33002 > 10.0.2.15.ssh: Flags [P.], seq 2653:2653, ac
k 3578, win 40880, length 0
^C
48 packets captured
48 packets received by filter
0 packets dropped by kernel
student@vml:~$
```