Homework #7

1. Do “clear”, then do “ps –ef”, thend do “ps –ef | grep init”.
    - Take a screenshot

```
mtak        6447    6340  0 01:49 pts/0    00:00:00 ps -ef
mtak@mtak-virtual-machine:~$ ps -ef | grep init
root           1       0  0  5월16 ?       00:00:04 /sbin/init auto noprompt splash
mtak        2491    2169  0  5월16 ?       00:00:02 /usr/bin/Xwayland :0 -rootless -noreset -accessx -c
ore -auth /run/user/1000/.mutter-Xwaylandauth.Y5HK41 -listen 4 -listen 5 -displayfd 6 -initfd 7
mtak        6449    6340  0 01:49 pts/0    00:00:00 grep --color=auto init
mtak@mtak-virtual-machine:~$
```

    - What is the program “grep”?
      grep 뒤의 문자열을 포함하는 결과만 출력한다.
    - What is the PID of “init” program?
      1이다.

2. Do “clear”, then do “top”, then press “space key”, then press “M, then press “P”, then press “q”.

- Take a screenshot

```
Tasks: 303 total,   1 running, 302 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.5 us,  0.4 sy,  0.0 ni, 99.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3888.8 total,    821.9 free,   1428.2 used,   1638.7 buff/cache
MiB Swap:   2140.0 total,   2140.0 free,      0.0 used.   2194.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   6455 mtak      20   0   23320   4552   3684 R   2.1   0.1   0:00.17 top
   2169 mtak      20   0 4603376 275036 133940 S   1.1   6.9   0:59.12 gnome-shell
   6059 root      20   0       0      0      0 I   1.1   0.0   0:14.20 kworker/0:3-events
    671 systemd+  20   0   14828   6180   5388 S   0.5   0.2   0:21.33 systemd-oomd
    755 root      20   0  327336   9572   7932 S   0.5   0.2   0:31.68 vmtoolsd
   2280 mtak      20   0  325508  13020   7412 S   0.5   0.3   0:02.85 ibus-daemon
   2364 mtak      20   0  298780  38604  29832 S   0.5   1.0   0:28.40 vmtoolsd
   6232 root      20   0       0      0      0 I   0.5   0.0   0:08.14 kworker/2:1-events
   6241 root      20   0       0      0      0 I   0.5   0.0   0:03.88 kworker/1:1-mm_percpu_wq
      1 root      20   0  167780  13104   8256 S   0.0   0.3   0:04.26 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.06 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthread
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.22 ksoftirqd/0
     15 root      20   0       0      0      0 I   0.0   0.0   0:06.34 rcu_preempt
     16 root      rt   0       0      0      0 S   0.0   0.0   0:00.27 migration/0
     17 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     21 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     22 root      rt   0       0      0      0 S   0.0   0.0   0:00.96 migration/1
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.20 ksoftirqd/1
     25 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kblockd
     26 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
     27 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/2
     28 root      rt   0       0      0      0 S   0.0   0.0   0:00.91 migration/2
     29 root      20   0       0      0      0 S   0.0   0.0   0:01.45 ksoftirqd/2
     31 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H-events_highpri
     32 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
     33 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 inet_frag_wq
     34 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kauditd
     38 root      20   0       0      0      0 S   0.0   0.0   0:00.02 khungtaskd
     40 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
     42 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 writeback
     43 root      20   0       0      0      0 S   0.0   0.0   0:01.20 kcompactd0
     44 root      25   5       0      0      0 S   0.0   0.0   0:00.00 ksmd
     45 root      39  19       0      0      0 S   0.0   0.0   0:00.00 khugepaged
     46 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kintegrityd
     47 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kblockd
     48 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 blkcg_punt_bio
     50 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 tpm_dev_wq
     51 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 ata_sff
     52 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 md
     53 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 edac-poller
     54 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 devfreq_wq
     55 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 watchdogd
     56 root       0 -20       0      0      0 I   0.0   0.0   0:00.19 kworker/2:1H-kblockd
     57 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kswapd0
     58 root      20   0       0      0      0 S   0.0   0.0   0:00.00 ecryptfs-kthread
     65 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kthrotld
     72 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/24-pciehp
     73 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/25-pciehp
     74 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/26-pciehp
     75 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/27-pciehp
     76 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/28-pciehp
     77 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/29-pciehp
mtak@mtak-virtual-machine:~$ S
```

- What is each key “space”, “M”, “P”, “q” for the top interactive program?

  q는 종료를 나타내고 다시 터미널로 돌아간다.

  M은 메모리 사용량 순으로 프로세스를 정렬한다.

  P는 CPU 사용량 순으로 프로세스를 정렬한다

3. Do “clear”, then do “ps u”, then do “bash”, then do “bash”, then do “bash”, then do “ps l”, then do “ps f”, then do “exit”, then do “exit”, then do “exit”.

- Take a screenshot

```
mtak@mtak-virtual-machine:~$ ps u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mtak        1989  0.0  0.1 172348  6248 tty2     Ssl+  5월16   0:00 /usr/libexec/gdm-wayland-session
mtak        1997  0.0  0.3 232992 15840 tty2     Sl+   5월16   0:00 /usr/libexec/gnome-session-binary
mtak        2922  0.0  0.1  21112  5832 pts/0    Ss    5월16   0:00 bash
mtak        6340  0.0  0.1  20980  5520 pts/0    S    01:30   0:00 bash
mtak        6471  0.0  0.0  22636  3816 pts/0    R+   01:53   0:00 ps u
mtak@mtak-virtual-machine:~$ bash
mtak@mtak-virtual-machine:~$ bash
mtak@mtak-virtual-machine:~$ bash
mtak@mtak-virtual-machine:~$ ps l
F   UID     PID    PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4  1000    1989    1849  20   0 172348  6248 do_pol Ssl+ tty2       0:00 /usr/libexec/gdm-wayland-sess
0  1000    1997    1989  20   0 232992 15840 do_pol Sl+  tty2       0:00 /usr/libexec/gnome-session-bi
0  1000    2922    2904  20   0  21112  5832 do_wai Ss   pts/0      0:00 bash
4  1000    6340    6339  20   0  20980  5520 do_wai S    pts/0      0:00 bash
0  1000    6472    6340  20   0  20980  5524 do_wai S    pts/0      0:00 bash
0  1000    6478    6472  20   0  20976  5524 do_wai S    pts/0      0:00 bash
0  1000    6484    6478  20   0  20980  5540 do_wai S    pts/0      0:00 bash
0  1000    6492    6484  20   0  22636  1564 -      R+   pts/0      0:00 ps l
mtak@mtak-virtual-machine:~$ ps f
    PID TTY      STAT   TIME COMMAND
   6340 pts/0    S      0:00 bash
   6472 pts/0    S      0:00  \_ bash
   6478 pts/0    S      0:00      \_ bash
   6484 pts/0    S      0:00          \_ bash
   6493 pts/0    R+     0:00              \_ ps f
   2922 pts/0    Ss     0:00 bash
   1989 tty2     Ssl+   0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL_SESSION_MODE=ubuntu /usr
   1997 tty2     Sl+    0:00  \_ /usr/libexec/gnome-session-binary --session=ubuntu
mtak@mtak-virtual-machine:~$ exit
exit
mtak@mtak-virtual-machine:~$ exit
exit
mtak@mtak-virtual-machine:~$ exit
exit
mtak@mtak-virtual-machine:~$
```

- Which process was the parent process of the process “ps f”?
  3번째 bash인 pid 6484

4. Do “clear”, then do “vi a”, then press “ctrl+z”, then do “jobs”, then do “sleep 10000&”, then do “jobs”, then do “vi b”, then press “ctrl+z”, then do “jobs”, then do “vi c”, then press “ctrl+z”, then do “jobs”, then do “fg”, then do “:q!”, then do “jobs”.

- Take a screenshot

```
mtak@mtak-virtual-machine:~$ vi a

[1]+  Stopped                 vi a
mtak@mtak-virtual-machine:~$ jobs
[1]+  Stopped                 vi a
mtak@mtak-virtual-machine:~$ sleep 10000&
[2] 6497
mtak@mtak-virtual-machine:~$ jobs
[1]+  Stopped                 vi a
[2]-  Running                 sleep 10000 &
mtak@mtak-virtual-machine:~$ vi b

[3]+  Stopped                 vi b
mtak@mtak-virtual-machine:~$ jobs
[1]-  Stopped                 vi a
[2]   Running                 sleep 10000 &
[3]+  Stopped                 vi b
mtak@mtak-virtual-machine:~$ vi c

[4]+  Stopped                 vi c
mtak@mtak-virtual-machine:~$ jobs
[1]   Stopped                 vi a
[2]   Running                 sleep 10000 &
[3]-  Stopped                 vi b
[4]+  Stopped                 vi c
mtak@mtak-virtual-machine:~$ fg
vi c
mtak@mtak-virtual-machine:~$ jobs
[1]-  Stopped                 vi a
[2]   Running                 sleep 10000 &
[3]+  Stopped                 vi b
mtak@mtak-virtual-machine:~$
```

- What do “-“ and “+” mean in the results of jobs program?

  + 는 가장 최근에 활성화한 작업이다.

  - 는 이전 현재 작업이다.

5. Do “clear”, then do “fg %1”, then do “:q!”, then do “jobs”, then do “fg”, then do “:q!”, then do “jobs”, then do “kill -9 $(ps | grep sleep | awk ‘{print $1}’)”, then press enter, then do “jobs”.

- Take a screenshot

```
mtak@mtak-virtual-machine: ~
mtak@mtak-virtual-machine:~$ fg %1
vi a
mtak@mtak-virtual-machine:~$ jobs
[2]-  Running                 sleep 10000 &
[3]+  Stopped                 vi b
mtak@mtak-virtual-machine:~$ fg
vi b
mtak@mtak-virtual-machine:~$ jobs
[2]+  Running                 sleep 10000 &
mtak@mtak-virtual-machine:~$ kill -9 $(ps | grep sleep | awk ‘{print $1}’)
awk: 1: unexpected character 0xe2
awk: line 2: missing } near end of file
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
mtak@mtak-virtual-machine:~$ kill -9 $(ps | grep sleep | awk {print $1})
awk: line 2: missing } near end of file
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
mtak@mtak-virtual-machine:~$ echo $1

mtak@mtak-virtual-machine:~$ print $1
mtak@mtak-virtual-machine:~$ kill -9 -$(ps | grep sleep | awk {print $1})
awk: line 2: missing } near end of file
bash: kill: -: arguments must be process or job IDs
mtak@mtak-virtual-machine:~$ kill -9 -$(ps | grep sleep | awk ‘{print $1}’)
awk: 1: unexpected character 0xe2
awk: line 2: missing } near end of file
bash: kill: -: arguments must be process or job IDs
mtak@mtak-virtual-machine:~$ jobs
[2]+  Running                 sleep 10000 &
mtak@mtak-virtual-machine:~$ S
```

- Describe the command “kill -9 $(ps | grep sleep | awk ‘{print $1}’)”.

- `ps` 명령어를 사용하여 현재 실행 중인 프로세스 목록을 표시합니다.
- `grep sleep`은 `ps` 결과에서 "sleep"이라는 단어를 포함하는 줄을 필터링합니다.
- `awk '{print $1}'`은 `grep` 결과에서 첫 번째 열을 출력합니다. 여기서는 프로세스 ID(PID)입니다.
- `$(...)`는 내부 명령어를 실행한 결과를 변수로 확장합니다.
- `kill -9`는 PID를 사용하여 해당 프로세스를 강제로 종료합니다. `-9`는 SIGKILL 시그널을 전달하여 강제 종료를 의미합니다.

6.
    1. Prepare a user “peterpan”. And make sure that “peterpan” has its home directory and become a sudoer. (Creating : #useradd –m –G sudo –s /bin/bash peterpan) (Setting Password : #passwd peterpan). Then switch user to peterpan and set pwd to peterpan’s home directory. (su – peterpan)
    2. Prepare “at” command. If “at” command is not installed, use “apt-get install at” to install it
    3. Do “clear”, then do “at now”, then type “/bin/ls –l > ~/ls.out”, then in the next line press “ctrl+d”. Then do “ls –l ls.out” and do “cat ls.out”
    4. Do “ echo ‘/bin/ls –l > ~/ls.out.1’ > ls.sh ”, then do “at now + 1 minutes < ls.sh” then do “atq”, wait around 1 minute do “atq” again, then do “ls –l ls.out*”

- Take a screenshot

```
peterpan@mtak-virtual-machine:~$ at now
warning: commands will be executed using /bin/sh
at Wed May 17 02:39:00 2023
at> /bin/ls -l > ~/ls.out
at> <EOT>
job 1 at Wed May 17 02:39:00 2023
peterpan@mtak-virtual-machine:~$ ls -l
total 40
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Desktop
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Documents
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Downloads
-rw-rw-r-- 1 peterpan peterpan  598  5월 17 02:40 ls.out
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Music
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Pictures
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Public
drwx------ 4 peterpan peterpan 4096  5월 17 02:36 snap
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Templates
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Videos
peterpan@mtak-virtual-machine:~$ ls -l ls.out
-rw-rw-r-- 1 peterpan peterpan 598  5월 17 02:40 ls.out
peterpan@mtak-virtual-machine:~$ cat ls.out
total 36
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Desktop
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Documents
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Downloads
-rw-rw-r-- 1 peterpan peterpan    0  5월 17 02:40 ls.out
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Music
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Pictures
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Public
drwx------ 4 peterpan peterpan 4096  5월 17 02:36 snap
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Templates
drwxr-xr-x 2 peterpan peterpan 4096  5월 17 02:35 Videos
peterpan@mtak-virtual-machine:~$ echo '/bin/ls -l > ~/ls.out.1' > ls.sh
peterpan@mtak-virtual-machine:~$ at now + 1 minutes < ls.sh
warning: commands will be executed using /bin/sh
job 2 at Wed May 17 02:44:00 2023
peterpan@mtak-virtual-machine:~$ atq
2       Wed May 17 02:44:00 2023 a peterpan
peterpan@mtak-virtual-machine:~$ atq
peterpan@mtak-virtual-machine:~$ ls -l ls.out
-rw-rw-r-- 1 peterpan peterpan 598  5월 17 02:40 ls.out
peterpan@mtak-virtual-machine:~$
```

- Describe the difference of results of two “atq” commands
- 작업을 02:44분에 echo 명령어를 쳤는데, 2:44분에는 남아있다가 45분이 되자 실행이 되었다.

  따라서 사진을 보면, 첫번째 atq에는 02:44 이 남아 있지만,02:45분이후에 친 atq 에서는 사라졌다. 따라서 at now +1 minutes 은 1분 뒤에 실행을 예약하는 명령어임을 알 수 있다.
- What is the meaning of “at now +1 minutes”?

  1분 뒤에 실행하라는 말이다.

5. Do “clear”, then do “at 9:00 tomorrow < ls.sh”, then do “at 13:30 19.06.04 < ls.sh”, then do “at 13:30 04.06.50 < ls.sh”, then do “at 8:00 + 3 days < ls.sh”, then do “at 17:00 + 3 weeks < ls.sh”, then do “atq”, then do “sudo ls –l /var/spool/cron/atjobs”.

- Take a screenshot

```
peterpan@mtak-virtual-machine:~$ at at 9:00 tomorrow < ls.sh
syntax error. Last token seen: a
Garbled time
peterpan@mtak-virtual-machine:~$ at 9:00 tomorrow < ls.sh
warning: commands will be executed using /bin/sh
job 3 at Thu May 18 09:00:00 2023
peterpan@mtak-virtual-machine:~$ at 13:30 19.06.04 < ls.sh
at: refusing to create job destined in the past
peterpan@mtak-virtual-machine:~$ at 13:30 04.06.50 < ls.sh
warning: commands will be executed using /bin/sh
job 4 at Sat Jun  4 13:30:00 2050
peterpan@mtak-virtual-machine:~$ at 8:00 + 3 days < ls.sh
warning: commands will be executed using /bin/sh
job 5 at Sat May 20 08:00:00 2023
peterpan@mtak-virtual-machine:~$ at 17:00 + 3 weeks < ls.sh
warning: commands will be executed using /bin/sh
job 6 at Wed Jun  7 17:00:00 2023
peterpan@mtak-virtual-machine:~$ atq
6       Wed Jun  7 17:00:00 2023 a peterpan
4       Sat Jun  4 13:30:00 2050 a peterpan
5       Sat May 20 08:00:00 2023 a peterpan
3       Thu May 18 09:00:00 2023 a peterpan
peterpan@mtak-virtual-machine:~$ sudo ls -l /var/spool/cron/atjobs
ls: cannot access '-l': No such file or directory
/var/spool/cron/atjobs:
a0000301ac5b60  a0000402856dce  a0000501ac6664  a0000601accdc0
peterpan@mtak-virtual-machine:~$
```

```
peterpan@mtak-virtual-machine:~$ sudo ls -l /var/spool/cron/atjobs
total 32
-rwx------ 1 peterpan daemon 4561  5월 17 02:49 a0000301ac5b60
-rwx------ 1 peterpan daemon 4561  5월 17 02:50 a0000402856dce
-rwx------ 1 peterpan daemon 4561  5월 17 02:50 a0000501ac6664
-rwx------ 1 peterpan daemon 4561  5월 17 02:50 a0000601accdc0
```

- How to remove the job which is scheduled on year of 2050?

  50년에 예약된 작업 번호는 9번이다. 따라서 **atrm 4** 로 지울 수 있다.

  혹은 **at -d 4** 로 지울 수 있다.

7. 1. Prepare a user “peterpan”. And make sure that “peterpan” has its home directory and become a sudoer. (Creating : #useradd –m –G sudo –s /bin/bash peterpan) (Setting Password : #passwd peterpan).
   Then switch user to peterpan and set pwd to peterpan’s home directory. (su – peterpan)
   2. Do “clear”, then do “sudo ls –l /var/spool/cron/crontabs”, then do “EDITOR=vi; export EDITOR”, then do “crontab –e”, then insert “0 5 * * * /bin/ps –ef > ~peterpan/ps.out” on the last line, then exit vi editor with saving the modification (“:wq”), then do “sudo ls –l /var/spool/cron/crontabs”, then do “sudo tail –3 /var/spool/cron/crontabs/peterpan”, then do “crontab –r”, then do “sudo ls –l /var/spool/cron/crontabs”

- Take a screenshot

```
peterpan@mtak-virtual-machine:~$ sudo ls -l /var/spool/cron/crontabs
ls: cannot access '-l': No such file or directory
/var/spool/cron/crontabs:
peterpan@mtak-virtual-machine:~$ sudo ls -l /var/spool/cron/crontabs
total 0
peterpan@mtak-virtual-machine:~$ EDITOR=vi; export EDITOR
peterpan@mtak-virtual-machine:~$ crontab -e
no crontab for peterpan - using an empty one
crontab: installing new crontab
peterpan@mtak-virtual-machine:~$ sudo ls -l /var/spool/cron/crontabs
total 4
-rw------- 1 peterpan crontab 1133  5월 17 02:59 peterpan
peterpan@mtak-virtual-machine:~$ sudo tail -3 /var/spool/cron/crontabs/peterpan
tail: cannot open '-3' for reading: No such file or directory
==> /var/spool/cron/crontabs/peterpan <==
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 5 * * * /bin/ps -ef > ~peterpan/ps.out
peterpan@mtak-virtual-machine:~$ crontab -r
peterpan@mtak-virtual-machine:~$ sudo ls -l  /var/spool/cron/crontabs
total 0
peterpan@mtak-virtual-machine:~$
```

- What is the meaning of “0 5 * * * /bin/ps –ef > ~peterpan/ps.out”?
  매 5시간 0분 마다 ~peterpan/ps.out에 대한 ps -ef 명령 실행.
- What is the functionality of “crontab –r”?
  crontab을 삭제한다.

3. Do “clear”, then do “sudo touch /etc/cron.deny”, the do “sudo vi /etc/cron.deny”, then insert “peterpan”, then exit vi editor with saving the modification (“:wq”), then do “crontab –e”, then do “sudo rm /etc/cron.deny”, then do “crontab –e”, then quit (“:q”) -

- Take a screenshot

```
peterpan@mtak-virtual-machine:~$ sudo touch /etc/cron.deny
peterpan@mtak-virtual-machine:~$ sudo vi /etc/cron.deny~
peterpan@mtak-virtual-machine:~$ crontab -e
no crontab for peterpan - using an empty one
crontab: installing new crontab
peterpan@mtak-virtual-machine:~$ sudo rm /etc/cron.deny
peterpan@mtak-virtual-machine:~$ crontab -e
No modification made
peterpan@mtak-virtual-machine:~$
```

- Describe your observation
  cron.deny 는 cron.deny에 이름을 등록하면 해당 사용자는 crontab을 사용하지 못하게 하는 블랙리스트 같은 기능이라고 생각한다. 생각과 같이, peterpan을 등록시킨 cron.deny가 있을 때는 허용되지 않고, cron.deny를 삭제하니 정상적으로 crontab -e 가 실행되었다.

Problems

1. Please execute following set of commands:

$ bash

$ sleep 100 => you need to stop this with “ctrl+z” to run “ps”

$ exec sleep 100 => you need to stop this with “ctrl+z” to run “ps”

Please check the information of processes for bash and sleep, and describe the difference between “sleep 100” and “exec sleep 100” command. (HINT: the functionality of exec”)

sleep 100은 fork()로 호출되게 된다. 하지만 exec sleep 100은 exec() 로 호출되게 되는데, fock는 프로세스가 하나 더 생기는 반면에, exec는 exec()를 호출한 프로세스의 PPID가 그대로 덮어 쓰여진다. 또, 이 차이에 따라 작업 이름도 다름을 알 수있다.

```
peterpan@mtak-virtual-machine:~$ bash
peterpan@mtak-virtual-machine:~$ sleep 100
^Z
[1]+  Stopped                 sleep 100
peterpan@mtak-virtual-machine:~$ ps
    PID TTY          TIME CMD
   3848 pts/0    00:00:00 bash
   5290 pts/0    00:00:00 bash
   5296 pts/0    00:00:00 sleep
   5297 pts/0    00:00:00 ps
peterpan@mtak-virtual-machine:~$ exec sleep 100
^Z
[1]+  Stopped                 bash
peterpan@mtak-virtual-machine:~$ ps
    PID TTY          TIME CMD
   3848 pts/0    00:00:00 bash
   5290 pts/0    00:00:00 sleep
   5296 pts/0    00:00:00 sleep <defunct>
   5298 pts/0    00:00:00 ps
peterpan@mtak-virtual-machine:~$
```

2. Setup your crontab to run “/bin/ps –ef > ~/ps.out”. Show the proof how you did this. (Hint : Spool, date, ls)

   crontab -e

   /bin/ps -ef > ~/ps.out 작성 및 저장.(**:wq**)

   sudo ls -l /var/spool/cron/crontabs **:** 크론탭 목록 보기

   sudo tail -3 /var/spool/cron/crontabs/mtak : 내 크론탭 파일 끝내용 보여줌

3. Setup the following jobs by using “crontab –e”. Then show the contents of your crontab
   1. Run “/bin/ps –ef > ~/ps.out” for every 10 and 40 minute of each hour
   2. Run “/bin/ps –ef > ~/ps.out” for every Monday morning 7:00
   3. Run “/bin/ps –ef > ~/ps.out” for every 2 minutes between 9:00 and 9:50 on the first day of each month

4. Run “/bin/ps –ef > ~ps.out” for every working day (Mon-Fri) 17:13

```
peterpan@mtak-virtual-machine: ~
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
10,40 * * * * /bin/ps -ef > ~/ps.out
0 7 * * 1 /bin/ps -ef > ~/ps.out
0-50/2 9 1 * * /bin/ps -ef > ~/ps.out
13 17 * * 1-5 /bin/ps -ef > ~/ps.out
~
```