



**LABORATORIUM PEMBELAJARAN ILMU KOMPUTER**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS BRAWIJAYA**

---

BAB : PROCESS DAN THREAD  
NAMA : DANI ADRIAN  
NIM : 225150201111009  
TANGGAL : 21/03/2023  
ASISTEN : ZHAFRAN RAMA AZMI  
GIBRAN HAKIM

---

## Langkah Praktikum

### 3.3.1 Proses

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan Start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1.

#### 3.3.1.1 Menjalankan perintah dasar Linux

Setelah berhasil login, lakukan perintah berikut ini, amati, buat rekaman screenshoot dan atau catat hasil yang diperoleh untuk mendapatkan hasil-hasil yang akan dibahas dibagian pembahasan.

##### 1. man ps

```
PS(1) User Commands PS(1)
NAME
  ps - report a snapshot of the current processes.

SYNOPSIS
  ps [options]

DESCRIPTION
  ps displays information about a selection of the active processes. If you want a repetitive update of the
  selection and the displayed information, use top(1) instead.

  This version of ps accepts several kinds of options:

  1  UNIX options, which may be grouped and must be preceded by a dash.
  2  BSD options, which may be grouped and must not be used with a dash.
  3  GNU long options, which are preceded by two dashes.

  Options of different types may be freely mixed, but conflicts can appear. The process selection options
  which are functionally identical, due to the many standards and ps implementations.

  Note that "ps -aux" is distinct from "ps aux". The POSIX and UNIX standards require that ps
  processes owned by a user named "x", as well as printing all processes owned by the user. The ps
  option. If the user named "x" does not exist, this ps may interpret this as a warning. This behavior
  is intended to aid in transitioning old scripts to the new standard.

  By default, ps selects all processes with the same effective user ID (EUID) as the user who invoked
  ps. It displays the process ID (PID), the process name (comm), the CPU time in seconds (ttime), the
  associated with the process (tname=TTY), the cumulated CPU time in seconds (ttime), the process
  executable name (ucmd=CMD). Output is unsorted by default.

  The use of BSD-style options will add process state (stat=STAT) to the default display and show the command
  args (args=COMMAND) instead of the executable name. You can override this with the PS_FORMAT environment
  variable. The use of BSD-style options will also change the process selection to include processes on other
  terminals (TTYS) that are owned by you; alternately, this may be described as setting the selection to be the
  set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects
  are not considered when options are described as being "identical" below, so -M will be considered identical
  to Z and so on.

  Except as described below, process selection options are additive. The default selection is discarded, and
  then the selected processes are added to the set of processes to be displayed. A process will thus be shown
  if it meets any of the given selection criteria.

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

##### 2. top

```
top - 12:56:29 up 15 min, 0 users, load average: 0.01, 0.04, 0.00
Tasks: 101 total, 1 running, 58 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.3 us, 0.3 sy, 0.0 ni, 98.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st
KiB Mem : 987912 total, 564708 free, 174584 used, 248620 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 674644 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 3401 ec2-user  20   0 717616 51592 32244 S  0.7   5.2   0:01.15 node
    1 root      20   0 41548  5256  3828 S  0.0   0.5   0:02.58 systemd
    2 root      20   0      0      0      0 S  0.0   0.0   0:00.00 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      20   0      0      0      0 I  0.0   0.0   0:00.00 kworker/0:0-eve
    6 root      0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/0:0H-ev
    7 root      20   0      0      0      0 I  0.0   0.0   0:00.02 kworker/u30:0-x
    8 root      0 -20   0      0      0 I  0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0      0      0      0 S  0.0   0.0   0:00.00 rcu_tasks_rude
   10 root      20   0      0      0      0 S  0.0   0.0   0:00.00 rcu_tasks_trace
   11 root      20   0      0      0      0 S  0.0   0.0   0:00.03 ksoftirqd/0
   12 root      20   0      0      0      0 I  0.0   0.0   0:00.09 rcu_sched
   13 root      rt    0      0      0      0 S  0.0   0.0   0:00.00 migration/0
   15 root      20   0      0      0      0 S  0.0   0.0   0:00.00 cpuhp/0

top - 12:56:47 up 15 min, 0 users, load average: 0.00, 0.03, 0.00
Tasks: 99 total, 2 running, 55 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 0.7 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 987912 total, 563708 free, 175548 used, 248656 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 673664 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 3273 ec2-user  20   0 875996 65276 34440 R  0.7   6.6   0:03.27 node
 3401 ec2-user  20   0 717616 52312 32244 S  0.7   5.3   0:01.24 node
 3211 ec2-user  20   0 151264  5036  3140 S  0.3   0.5   0:00.21 sshd
    1 root      20   0 41548  5256  3828 S  0.0   0.5   0:02.58 systemd
    2 root      20   0      0      0      0 S  0.0   0.0   0:00.00 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      20   0      0      0      0 I  0.0   0.0   0:00.00 kworker/0:0-eve
    6 root      0 -20   0      0      0 I  0.0   0.0   0:00.00 kworker/0:0H-ev
    7 root      20   0      0      0      0 I  0.0   0.0   0:00.02 kworker/u30:0-f
    8 root      0 -20   0      0      0 I  0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0      0      0      0 S  0.0   0.0   0:00.00 rcu_tasks_rude
   10 root      20   0      0      0      0 S  0.0   0.0   0:00.00 rcu_tasks_trace
   11 root      20   0      0      0      0 S  0.0   0.0   0:00.03 ksoftirqd/0
   12 root      20   0      0      0      0 I  0.0   0.0   0:00.09 rcu_sched
   13 root      rt    0      0      0      0 S  0.0   0.0   0:00.00 migration/0
```

config Dani Adrian.t x +

File Edit View

Dani Adrian  
225150201111009

Ln 1, Col 1 100% Windows (CRLF) UTF-8

### 3. ps

```
[ec2-user@ip-172-31-31-146 ~]$ ps
  PID TTY          TIME CMD
 3482 pts/4    00:00:00 bash
 4046 pts/4    00:00:00 ps
[ec2-user@ip-172-31-31-146 ~]$
```

config Dani Adrian.t • +

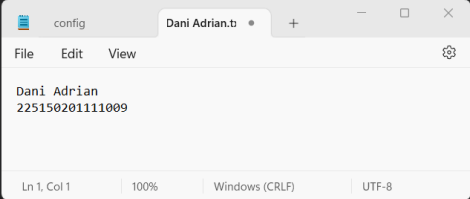
File Edit View

Dani Adrian  
225150201111009

Ln 1, Col 1 100% Windows (CRLF) UTF-8

### 4. ps -ax

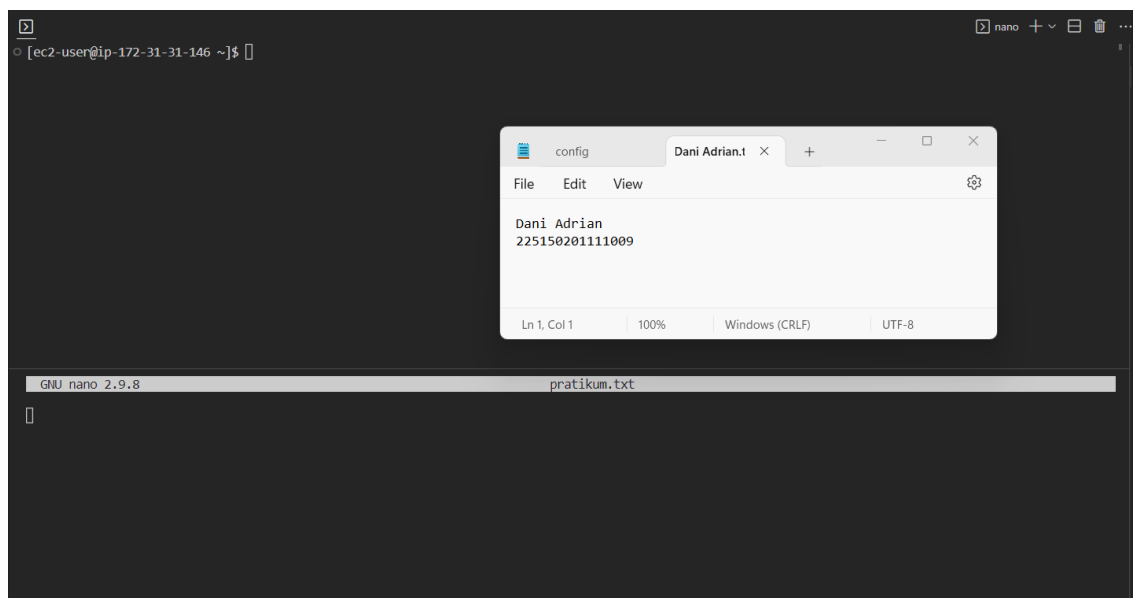
```
[ec2-user@ip-172-31-31-146 ~]$ ps -ax
PID TTY STAT TIME COMMAND
1 ? Ss 0:02 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
2 ? S 0:00 [kthreadd]
3 ? Ic 0:00 [rcu_gp]
4 ? Ic 0:00 [rcu_par_gp]
5 ? I 0:00 [kworker/0:0-eve]
6 ? Ic 0:00 [kworker/0:0H-ev]
7 ? I 0:00 [kworker/u30:0-e]
8 ? Ic 0:00 [mm_percpu_wq]
9 ? S 0:00 [rcu_tasks_rude_]
10 ? S 0:00 [rcu_tasks_trace]
11 ? S 0:00 [ksoftirqd/0]
12 ? I 0:00 [rcu_sched]
13 ? S 0:00 [migration/0]
15 ? S 0:00 [cpuhp/0]
17 ? S 0:00 [kdevtmpfs]
18 ? Ic 0:00 [netns]
21 ? S 0:00 [kauditd]
296 ? S 0:00 [khungtaskd]
297 ? S 0:00 [oom_reaper]
298 ? Ic 0:00 [writeback]
300 ? S 0:00 [kcompactd0]
301 ? SW 0:00 [ksmd]
302 ? SW 0:00 [khugepaged]
357 ? Ic 0:00 [kintegrityd]
359 ? Ic 0:00 [kblockd]
360 ? Ic 0:00 [blkcg_nunt_bio]
710 ? S 0:00 [xen-balloon]
711 ? I 0:00 [kworker/u30:2-x]
719 ? Ic 0:00 [tpm_dev_wq]
725 ? Ic 0:00 [md]
728 ? Ic 0:00 [edac-poller]
733 ? S 0:00 [watchdogd]
831 ? Ic 0:00 [kworker/0:1H-xf]
883 ? S 0:00 [kswapd0]
885 ? Ic 0:00 [xfsailloc]
886 ? Ic 0:00 [xfs_mru_cache]
880 ? Ic 0:00 [kthrotld]
900 ? S 0:00 [xenbus]
901 ? S 0:00 [xenwatch]
937 ? Ic 0:00 [nvme-wq]
939 ? Ic 0:00 [nvme-reset-wq]
941 ? Ic 0:00 [nvme-delete-wq]
974 ? Ic 0:00 [ipv6_addrconf]
```



5. Untuk melanjutkan praktikum, silakan buka satu koneksi ssh lagi ke server aws dengan cara yang sama dengan Langkah sebelumnya.

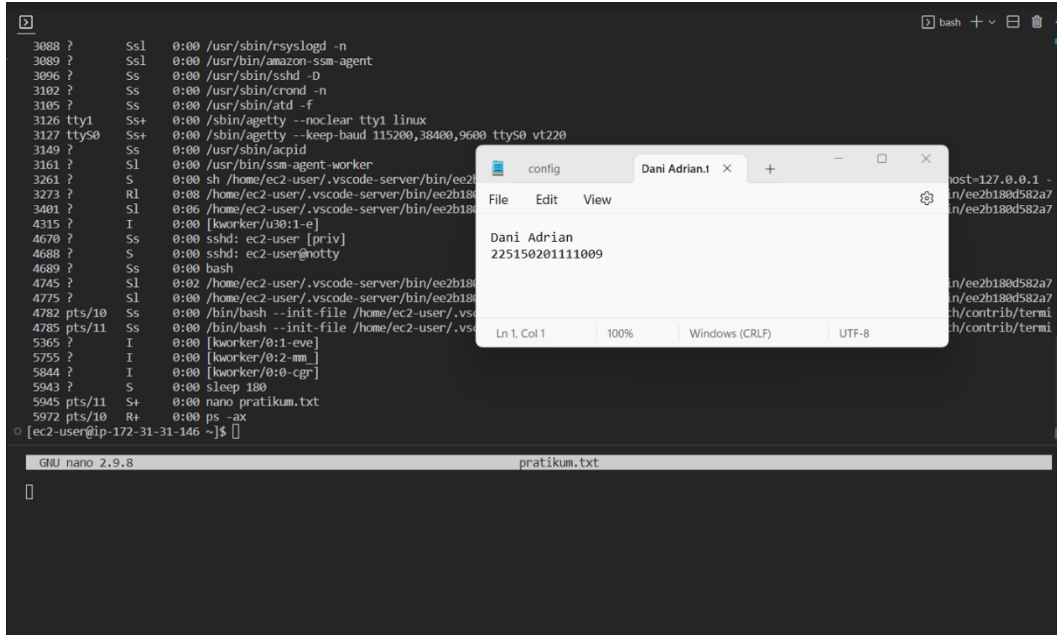
Terminal 1 adalah yang digunakan untuk menjalankan Langkah 1 – 4.  
Terminal 2 adalah yang dibuka pada Langkah ke 5

6. Pada terminal 2, bukalah file praktikum.txt dengan perintah nano praktikum.txt



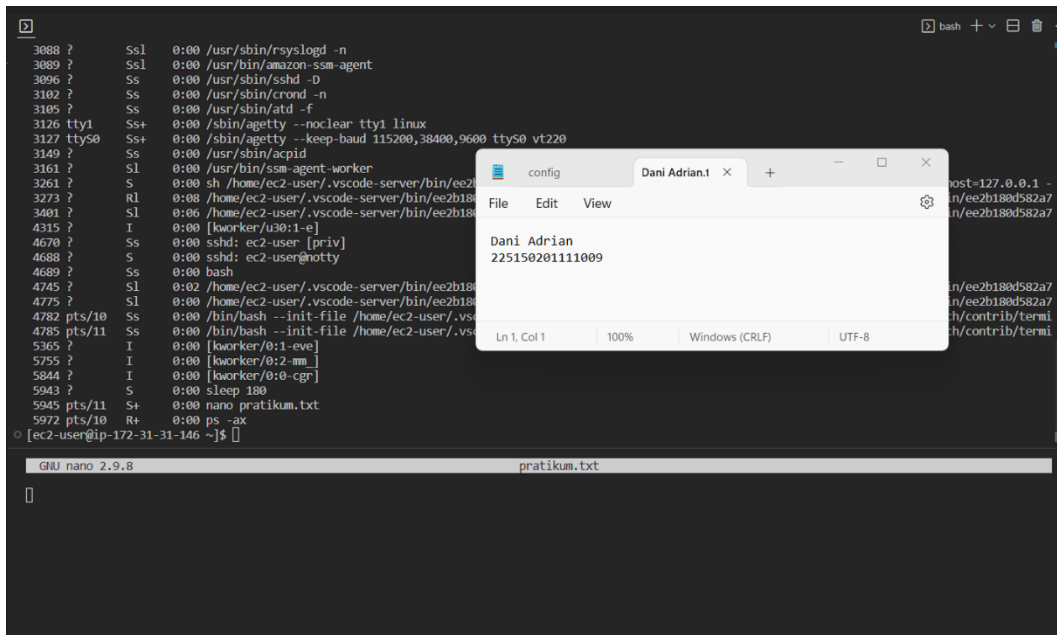
7. Pada terminal 1, lakukan aktifitas berikut ini:

a. Jalankan perintah `ps -ax`



```
3088 ?      Ssl      0:00 /usr/sbin/rsyslogd -n
3089 ?      Ssl      0:00 /usr/bin/amazon-ssm-agent
3096 ?      Ss       0:00 /usr/sbin/sshd -D
3102 ?      Ss       0:00 /usr/sbin/crond -n
3105 ?      Ss       0:00 /usr/sbin/atd -f
3126 tty1    Ss+      0:00 /sbin/agetty --noclear tty1 linux
3127 ttyS0    Ss+      0:00 /sbin/agetty --keep-baud 115200,38400,9600 ttyS0 vt220
3149 ?      Ss       0:00 /usr/sbin/acpid
3161 ?      Sl       0:00 /usr/bin/ssm-agent-worker
3261 ?      S        0:00 sh /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
3273 ?      Rl       0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
3401 ?      Sl       0:06 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4315 ?      I        0:00 [kworker/u30:1-e]
4670 ?      Ss       0:00 sshd: ec2-user [priv]
4688 ?      S        0:00 sshd: ec2-user@notty
4689 ?      Ss       0:00 bash
4745 ?      Sl       0:02 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4775 ?      Sl       0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4782 pts/10   Ss       0:00 /bin/bash --init-file /home/ec2-user/.vscode-...
4785 pts/11   Ss       0:00 /bin/bash --init-file /home/ec2-user/.vscode-...
5365 ?      I        0:00 [kworker/0:1-eve]
5755 ?      I        0:00 [kworker/0:2-mm_]
5844 ?      I        0:00 [kworker/0:0-cgr]
5943 ?      S        0:00 sleep 180
5945 pts/11   S+       0:00 nano pratikum.txt
5972 pts/10   R+       0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$
```

b. Carilah proses ID yang menjalankan perintah `nano`, gunakan proses ID tsb untuk menjalankan perintah (c)



```
3088 ?      Ssl      0:00 /usr/sbin/rsyslogd -n
3089 ?      Ssl      0:00 /usr/bin/amazon-ssm-agent
3096 ?      Ss       0:00 /usr/sbin/sshd -D
3102 ?      Ss       0:00 /usr/sbin/crond -n
3105 ?      Ss       0:00 /usr/sbin/atd -f
3126 tty1    Ss+      0:00 /sbin/agetty --noclear tty1 linux
3127 ttyS0    Ss+      0:00 /sbin/agetty --keep-baud 115200,38400,9600 ttyS0 vt220
3149 ?      Ss       0:00 /usr/sbin/acpid
3161 ?      Sl       0:00 /usr/bin/ssm-agent-worker
3261 ?      S        0:00 sh /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
3273 ?      Rl       0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
3401 ?      Sl       0:06 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4315 ?      I        0:00 [kworker/u30:1-e]
4670 ?      Ss       0:00 sshd: ec2-user [priv]
4688 ?      S        0:00 sshd: ec2-user@notty
4689 ?      Ss       0:00 bash
4745 ?      Sl       0:02 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4775 ?      Sl       0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7...
4782 pts/10   Ss       0:00 /bin/bash --init-file /home/ec2-user/.vscode-...
4785 pts/11   Ss       0:00 /bin/bash --init-file /home/ec2-user/.vscode-...
5365 ?      I        0:00 [kworker/0:1-eve]
5755 ?      I        0:00 [kworker/0:2-mm_]
5844 ?      I        0:00 [kworker/0:0-cgr]
5943 ?      S        0:00 sleep 180
5945 pts/11   S+       0:00 nano pratikum.txt
5972 pts/10   R+       0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$
```

c. Kemudian jalankan perintah `kill -9 [prosesID]`

```
3089 ? Ssl 0:00 /usr/bin/amazon-ssm-agent
3096 ? Ss 0:00 /usr/sbin/sshd -D
3102 ? Ss 0:00 /usr/sbin/crond -n
3105 ? Ss 0:00 /usr/sbin/atd -f
3126 tty1 Sst+ 0:00 /sbin/agetty --noclear tty1 linux
3127 ttys0 Sst+ 0:00 /sbin/agetty --keep-baud 115200,38400,9600 ttys0 vt220
3149 ? Ss 0:00 /usr/sbin/acpid
3161 ? Sl 0:00 /usr/bin/ssh-agent-worker
3261 ? S 0:00 sh /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
3273 ? Rl 0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
3401 ? Sl 0:06 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4315 ? I 0:00 [kworker/u30:1-e]
4670 ? Ss 0:00 sshd: ec2-user [priv]
4688 ? S 0:00 sshd: ec2-user@notty
4689 ? Ss 0:00 bash
4745 ? Sl 0:02 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4775 ? Sl 0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4782 pts/10 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4785 pts/11 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
5365 ? I 0:00 [kworker/0:1-eve]
5755 ? I 0:00 [kworker/0:2-mm_]
5844 ? I 0:00 [kworker/0:0-cgr]
5943 ? S 0:00 sleep 180
5945 pts/11 St+ 0:00 nano pratikum.txt
5972 pts/10 Rt+ 0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$ kill -9 5945
[ec2-user@ip-172-31-31-146 ~]$

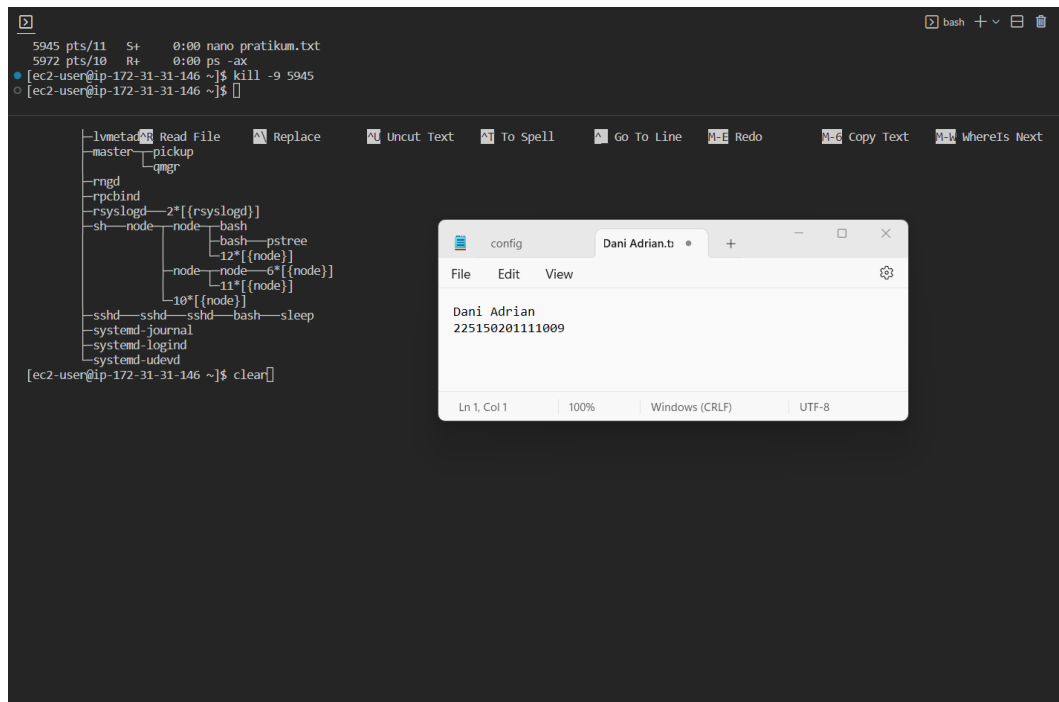
GNU nano 2.9.8 pratikum.txt
Killed
[ec2-user@ip-172-31-31-146 ~]$
```

8. Pada terminal 2, amati dan catat apa yang terjadi

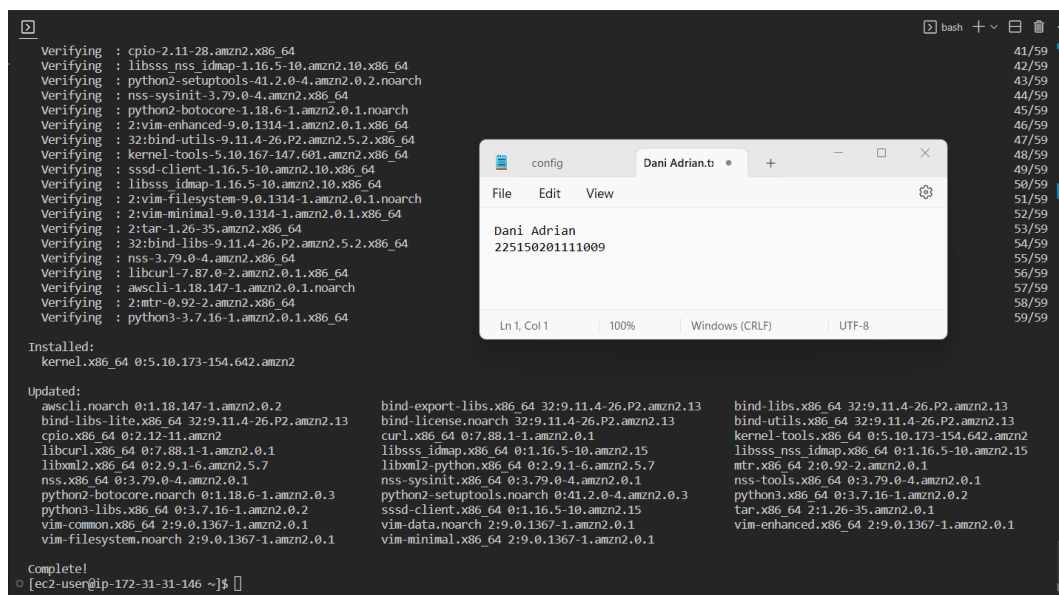
```
3089 ? Ssl 0:00 /usr/bin/amazon-ssm-agent
3096 ? Ss 0:00 /usr/sbin/sshd -D
3102 ? Ss 0:00 /usr/sbin/crond -n
3105 ? Ss 0:00 /usr/sbin/atd -f
3126 tty1 Sst+ 0:00 /sbin/agetty --noclear tty1 linux
3127 ttys0 Sst+ 0:00 /sbin/agetty --keep-baud 115200,38400,9600 ttys0 vt220
3149 ? Ss 0:00 /usr/sbin/acpid
3161 ? Sl 0:00 /usr/bin/ssh-agent-worker
3261 ? S 0:00 sh /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
3273 ? Rl 0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
3401 ? Sl 0:06 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4315 ? I 0:00 [kworker/u30:1-e]
4670 ? Ss 0:00 sshd: ec2-user [priv]
4688 ? S 0:00 sshd: ec2-user@notty
4689 ? Ss 0:00 bash
4745 ? Sl 0:02 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4775 ? Sl 0:00 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4782 pts/10 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
4785 pts/11 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0.0.1 -
5365 ? I 0:00 [kworker/0:1-eve]
5755 ? I 0:00 [kworker/0:2-mm_]
5844 ? I 0:00 [kworker/0:0-cgr]
5943 ? S 0:00 sleep 180
5945 pts/11 St+ 0:00 nano pratikum.txt
5972 pts/10 Rt+ 0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$ kill -9 5945
[ec2-user@ip-172-31-31-146 ~]$

GNU nano 2.9.8 pratikum.txt
Killed
[ec2-user@ip-172-31-31-146 ~]$
```

9. Pada terminal 2 jalankan perintah pstree



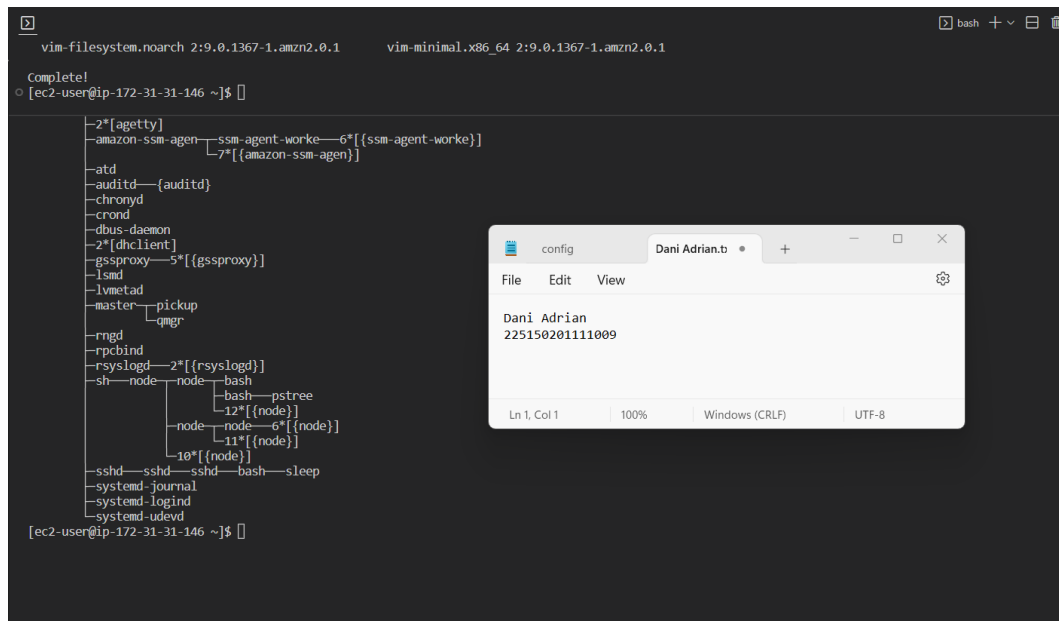
## 10. Jalankan perintah apt-get update pada terminal 1



## 11. Pada terminal 2, jalankan perintah pstree.

```
vim-filesystem.noarch 2:9.0.1367-1.amzn2.0.1 vim-minimal.x86_64 2:9.0.1367-1.amzn2.0.1
Complete!
[ec2-user@ip-172-31-31-146 ~]$ []

-2*[agetty]
-amazon-ssm-agent-ssm-agent-work-6*[{ssm-agent-work}]
-7*[{amazon-ssm-agent}]
-atd
-auditd-{auditd}
-chronyd
-cron
-dbus-daemon
-2*[dhclient]
-gssproxy-5*[{gssproxy}]
-lsm
-lvm2
-master-pickup
-qmgr
-rngd
-rpcbind
-rsyslogd-2*[{rsyslogd}]
-sh-node-node-bash
-bash-pstree
-12*[{node}]
-node-node-6*[{node}]
-11*[{node}]
-10*[{node}]
-sshd-sshd-sshd-bash-sleep
-systemd-journald
-systemd-logind
-systemd-udev
```



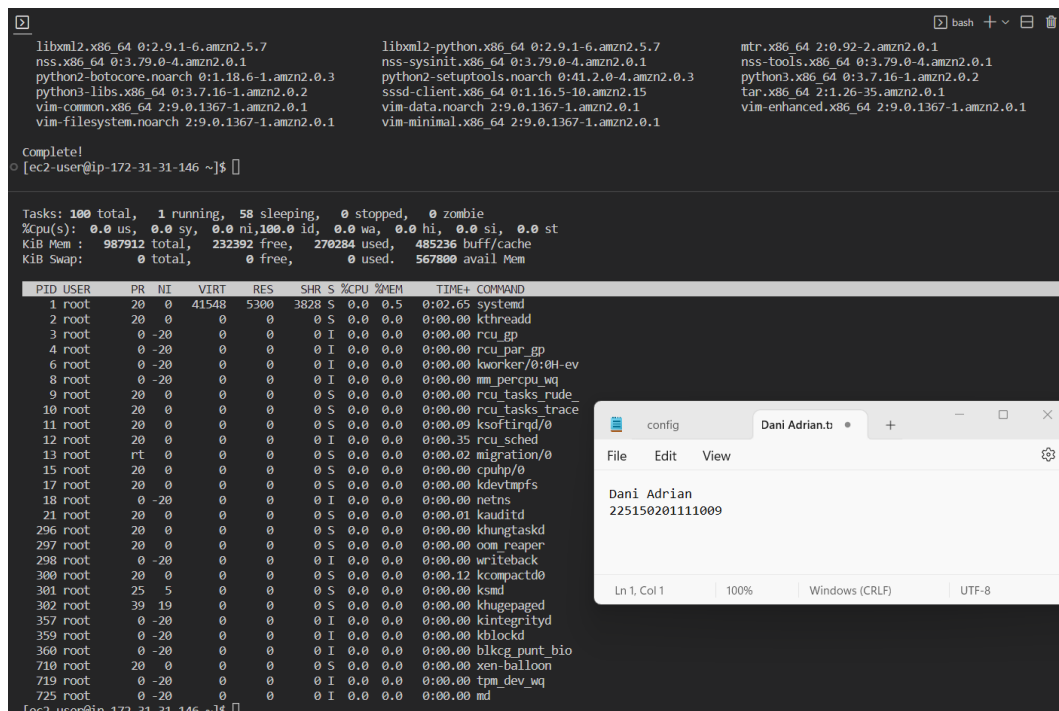
12. Kemudian masih pada terminal 2, Jalankan perintah top. Setelah perintah apt get update selesai pada terminal 1, hentikan perintah top dengan CTRL+C.

```
libxml2.x86_64 0:2.9.1-6.amzn2.5.7 libxml2-python.x86_64 0:2.9.1-6.amzn2.5.7 mtr.x86_64 2:0.92-2.amzn2.0.1
nss.x86_64 0:3.79.0-4.amzn2.0.1 nss-sysinit.x86_64 0:3.79.0-4.amzn2.0.1 nss-tools.x86_64 0:3.79.0-4.amzn2.0.1
python2-boto.noarch 0:1.18.6-1.amzn2.0.3 python2-setuptools.noarch 0:41.2.0-4.amzn2.0.3 python3.x86_64 0:3.7.16-1.amzn2.0.2
python3-libs.x86_64 0:3.7.16-1.amzn2.0.2 sssd-client.x86_64 0:1.16.5-10.amzn2.15 tar.x86_64 2:1.26-35.amzn2.0.1
vim-common.x86_64 2:9.0.1367-1.amzn2.0.1 vim-data.noarch 2:9.0.1367-1.amzn2.0.1 vim-enhanced.x86_64 2:9.0.1367-1.amzn2.0.1
vim-filesystem.noarch 2:9.0.1367-1.amzn2.0.1 vim-minimal.x86_64 2:9.0.1367-1.amzn2.0.1
```

Complete!  
[ec2-user@ip-172-31-31-146 ~]\$ []

Tasks: 100 total, 1 running, 58 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 : 987912 total, 232392 free, 270284 used, 485236 buff/cache  
KiB Swap: 0 total, 0 free, 0 used, 567800 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	41548	5300	3828	S	0.0	0.5	0:02.65	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	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
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-ev
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
11	root	20	0	0	0	0	S	0.0	0.0	0:00.09	ksoftirqd/0
12	root	20	0	0	0	0	I	0.0	0.0	0:00.35	rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
21	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kauditd
296	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
297	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
298	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
300	root	20	0	0	0	0	S	0.0	0.0	0:00.12	kcompactd0
301	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
302	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepaged
357	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kin integrityd
359	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
360	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
710	root	20	0	0	0	0	S	0.0	0.0	0:00.00	xen-balloon
719	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
725	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	md



13. Jalankan perintah pstree lagi pada terminal 2.

```
libxml2.x86_64 0:2.9.1-6.amzn2.5.7
nss.x86_64 0:3.79.0-4.amzn2.0.1
python2-boto.noarch 0:1.18.6-1.amzn2.0.3
python3-libs.x86_64 0:3.7.16-1.amzn2.0.2
vim-common.x86_64 2:9.0.1367-1.amzn2.0.1
vimfilesystem.noarch 2:9.0.1367-1.amzn2.0.1

libxml2-python.x86_64 0:2.9.1-6.amzn2.5.7
nss-sysinit.x86_64 0:3.79.0-4.amzn2.0.1
python2-setuptools.noarch 0:41.2.0-4.amzn2.0.3
sssd-client.x86_64 0:1.16.5-10.amzn2.15
vim-data.noarch 2:9.0.1367-1.amzn2.0.1
vim-minimal.x86_64 2:9.0.1367-1.amzn2.0.1

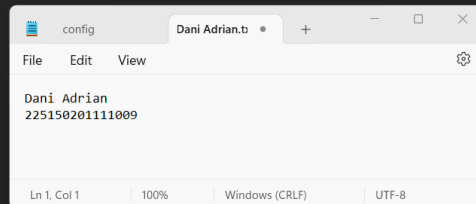
mtr.x86_64 2:0.92-2.amzn2.0.1
nss-tools.x86_64 0:3.79.0-4.amzn2.0.1
python3.x86_64 0:3.7.16-1.amzn2.0.2
tar.x86_64 2:1.26-35.amzn2.0.1
vim-enhanced.x86_64 2:9.0.1367-1.amzn2.0.1

Complete!
[ec2-user@ip-172-31-31-146 ~]$

710 root      20    0      0      0      0 S  0.0  0.0  0:00.00 xen-balloon
719 root      0 -20    0      0      0 I  0.0  0.0  0:00.00 tpm_dev_wq
725 root      0 -20    0      0      0 I  0.0  0.0  0:00.00 md

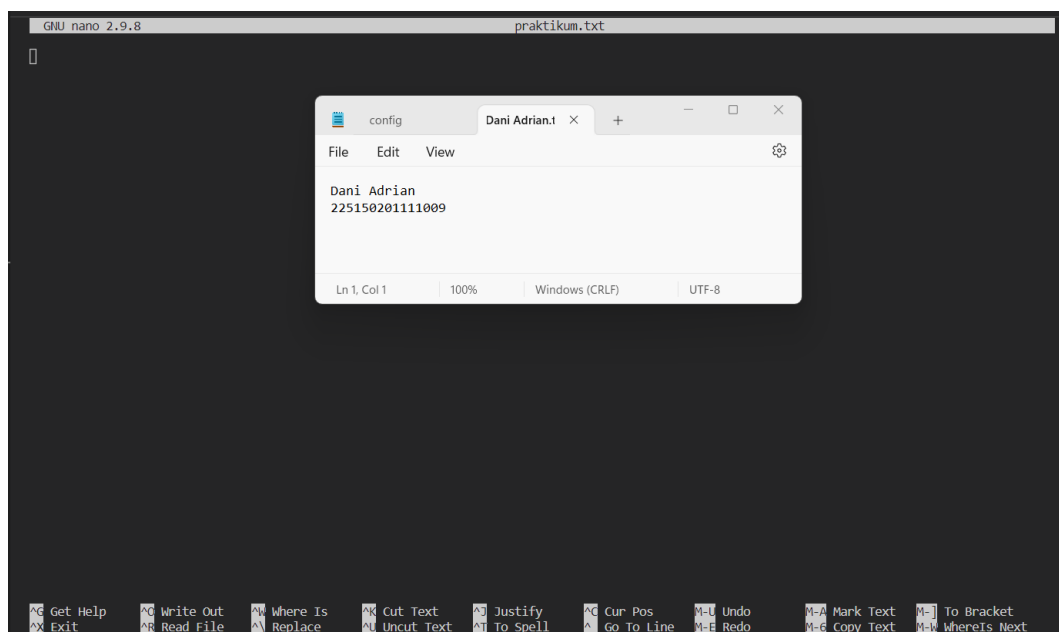
[ec2-user@ip-172-31-31-146 ~]$ pstree
systemd--acpid
          |
          +--2*[agetty]
          |
          +--amazon-ssm-agent--ssm-agent-worker--6*[ssm-agent-worker]
          |
          +--atd
          |
          +--auditd--{auditd}
          |
          +--chronyd
          |
          +--cron
          |
          +--dbus-daemon
          |
          +--2*[dhclient]
          |
          +--gssproxy--5*[gssproxy]
          |
          +--lsmd
          |
          +--lvm2metad
          |
          +--master--pickup
          |
          |   +--qmgr
          |
          +--rngd
          |
          +--rpcbind
          |
          +--rsyslogd--2*[rsyslogd]
          |
          +--sh--node--node--bash
          |
          |   +--node--bash--pstree
          |   |
          |   +--node--cpuUsage.sh--sleep
          |   |
          |   +--12*[node]
          |   |
          |   +--node--node--6*[node]
          |   |
          |   +--11*[node]
          |   |
          |   +--10*[node]
          |
          +--sshd--sshd--sshd--bash--sleep
          |
          +--systemd-journal
          |
          +--systemd-logind
          |
          +--systemd-udev

[ec2-user@ip-172-31-31-146 ~]$
```



### 3.3.1.2 Penggunaan Systemcall fork() dan exec()

#### 1. Pada terminal 1 jalankan program nano



#### 2. Tuliskan kode program berikut ini

```
#include <stdio.h>
#include <unistd.h>
/* This program forks and and the prints whether
the process is
* - the child (the return value of fork() is 0), or
```



```

* - the parent (the return value of fork() is not
zero)
*
* When this was run 100 times on the computer the
author is
* on, only twice did the parent process execute
before the
* child process executed.
*
* Note, if you juxtapose two strings, the compiler
automatically
* concatenates the two, e.g., "Hello " "world!"
*/
int main( void ) {
    int pid = fork();

    if ( pid == 0 ) {
        printf( "This is being printed from the
child process\n" );
    } else {
        printf( "This is being printed in the
parent process:\n- the process identifier (pid) of
the child is %d\n", pid );
    }

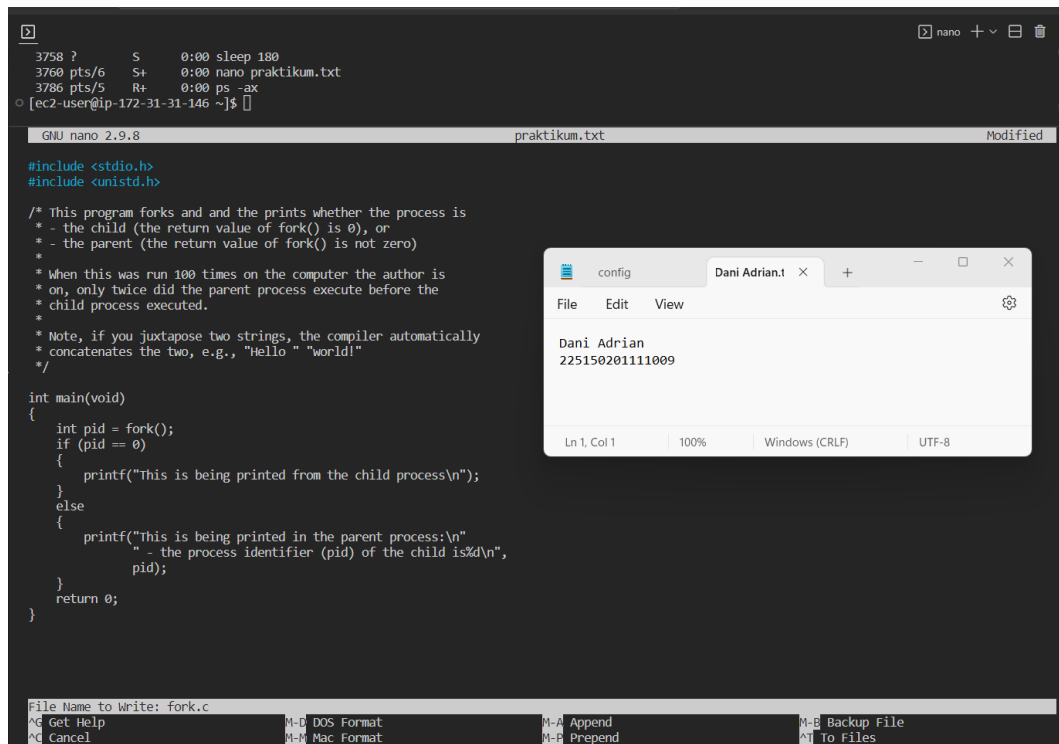
    return 0;
}

```

The screenshot shows a terminal window with the nano text editor open. The editor displays a C program that uses the `fork()` system call to create a child process. The program prints messages from both the parent and child processes. The terminal output shows the program running successfully, with the parent process printing the child's PID and the child process printing its own PID (0).

Below the terminal window, there is a secondary window titled "Dani Adrian.t" which displays the name "Dani Adrian" and a long alphanumeric string "225150201111009". This window appears to be a simple text editor or a window for displaying user information.

### 3. Save dengan nama fork.c



```
3758 ?      S      0:00 sleep 180
3760 pts/6    S+      0:00 nano praktikum.txt
3786 pts/5    R+      0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$
```

GNU nano 2.9.8 praktikum.txt Modified

```
#include <stdio.h>
#include <unistd.h>

/* This program forks and and the prints whether the process is
 * - the child (the return value of fork() is 0), or
 * - the parent (the return value of fork() is not zero)
 *
 * When this was run 100 times on the computer the author is
 * on, only twice did the parent process execute before the
 * child process executed.
 *
 * Note, if you juxtapose two strings, the compiler automatically
 * concatenates the two, e.g., "Hello " "world!"
 */

int main(void)
{
    int pid = fork();
    if (pid == 0)
    {
        printf("This is being printed from the child process\n");
    }
    else
    {
        printf("This is being printed in the parent process:\n"
               " - the process identifier (pid) of the child is%d\n",
               pid);
    }
    return 0;
}
```

File Name to Write: fork.c

^G Get Help ^M DOS Format ^M-A Append ^M-B Backup File  
^C Cancel ^M-M Mac Format ^M-P Prepend ^N To Files

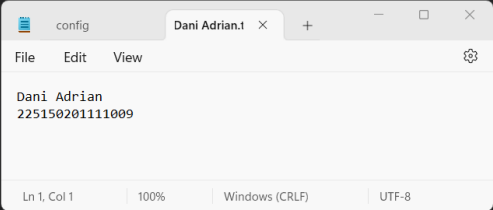
Dani Adrian  
2251502011111009

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

4. Keluar dari nano, Kembali ke terminal. Jalankan perintah gcc fork.c
5. Jika pada mesin EC2 belum dapat menjalankan perintah gcc, jalankan perintah berikut ini  
  
sudo apt-get install gcc
6. Ulangi jalankan perintah gcc fork.c

```
3211 ? S 0:00 sshd: ec2-user@notty
3212 ? Ss 0:00 bash
3261 ? S 0:00 sh /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/bin/code-server --start-server --host=127.0
3273 ? Sl 0:02 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/node /home/ec2-user/.vscode-server/bin/ee2b180
3407 ? Rl 0:01 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/node /home/ec2-user/.vscode-server/bin/ee2b180
3439 ? Sl 0:01 /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/node /home/ec2-user/.vscode-server/bin/ee2b180
3467 pts/4 Ss+ 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/out/vs/workbench/contrib
3523 pts/5 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/out/vs/workbench/contrib
3563 pts/6 Ss 0:00 /bin/bash --init-file /home/ec2-user/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8d9fd269ab1884/out/vs/workbench/contrib
3755 ? I 0:00 [kworker/0:0-mm]
3758 ? S 0:00 sleep 180
3760 pts/6 S+ 0:00 nano praktikum.txt
3786 pts/5 R+ 0:00 ps -ax
[ec2-user@ip-172-31-31-146 ~]$

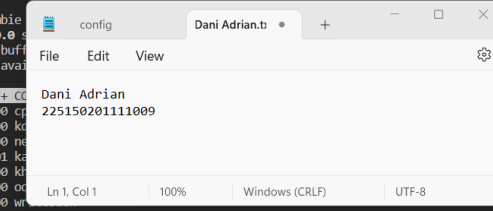
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ gcc fork.c
[ec2-user@ip-172-31-31-146 ~]$
```



## 7. Jalankan perintah ./a.out

```
libquadmath.x86_64 0:7.3.1-15.amzn2 libsanitizer.x86_64 0:7.3.1-15.amzn2 mpfr.x86_64 0:3.1.1-4.amzn2.0.2

complete!
[ec2-user@ip-172-31-31-146 ~]$ gcc fork.c
[ec2-user@ip-172-31-31-146 ~]$ ./a.out
This is being printed in the parent process:
- the process identifier (pid) of the child is11359
This is being printed from the child process
[ec2-user@ip-172-31-31-146 ~]$
```

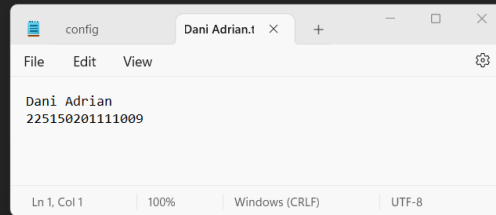


```
Tasks: 122 total, 1 running, 78 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.3 sy, 0.0 ni, 97.7 id, 0.3 wa, 0.0 hi, 0.0 st
K18 Mem : 987912 total, 251820 free, 291844 used, 444248 buff
K18 Swap: 0 total, 0 free, 0 used, 547724 avail

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ CO
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cp
17 root 20 0 0 0 0 S 0.0 0.0 0:00.00 k
18 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ne
21 root 20 0 0 0 0 S 0.0 0.0 0:00.01 ka
296 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kl
297 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oc
298 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 wr
300 root 20 0 0 0 0 S 0.0 0.0 0:00.18 kcompactd
301 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd
302 root 39 19 0 0 0 S 0.0 0.0 0:00.00 khugepaged
357 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kintegrityd
359 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kblockd
360 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 blkcg_punt_bio
710 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xen-balloon
719 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 tpm_dev_wq
725 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 md
728 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 edac-poller
733 root -51 0 0 0 0 S 0.0 0.0 0:00.00 watchdogd
831 root 0 -20 0 0 0 I 0.0 0.0 0:00.16 kworker/0:1H-kb
883 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kswapd0
885 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 xfsalloc
886 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 xfs_mru_cache
889 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kthrotld
900 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenbus
901 root 20 0 0 0 0 S 0.0 0.0 0:00.02 xenwatch
937 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 nvme-wq
939 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 nvme-reset-wq
```

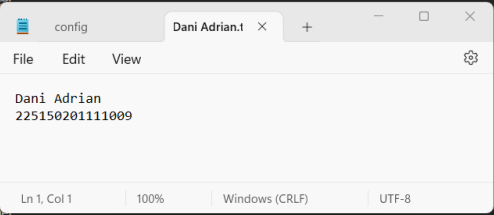
## 8. Sementara pada terminal 2 jalankan perintah pstree dan top

```
[ec2-user@ip-172-31-31-146 ~]$ pstree
systemd--acpid
      2*[agetty]
      amazon-ssm-agent--ssm-agent-worker--6*[{ssm-agent-worker}]
      --7*[{amazon-ssm-agent}]
      --anacron
      --atd
      auditd--{auditd}
      --chronyd
      --crond
      --dbus-daemon
      2*[dhclient]
      --gssproxy--5*[{gssproxy}]
      --lsm
      --lvm2metad
      --master--pickup
      --qmgr
      --rngd
      --rpcbind
      --rsyslogd--2*[{rsyslogd}]
      --sshd--sshd--sshd--bash--sh--node--node--2*[bash]
      --node--node--13*[{node}]
      --node--11*[{node}]
      --10*[{node}]
      --sleep
      --systemd-journal
      --systemd-logind
      --systemd-udev
```



```
top - 05:18:26 up 34 min, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 102 total, 1 running, 59 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.7 sy, 0.0 ni, 99.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 975604 total, 513232 free, 185860 used, 276512 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 650280 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 3273 ec2-user   20   0  880268  70088  34400 S   0.3   7.2   0:04.33 node
 3407 ec2-user   20   0  744520  56004  32396 S   0.3   5.7   0:03.65 node
 4312 ec2-user   20   0 168964   4464  3792 R   0.3   0.5   0:00.01 top
    1 root      20   0  41568   5188  3796 S   0.0   0.5   0:01.85 systemd
    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 rcu_gp
    4 root      20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_par_gp
    6 root      20   0      0      0      0 S   0.0   0.0   0:00.00 kworker/
    8 root      20   0      0      0      0 S   0.0   0.0   0:00.00 mm_perc
    9 root      20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_task
   10 root      20   0      0      0      0 S   0.0   0.0   0:00.00 rcu_task
   11 root      20   0      0      0      0 S   0.0   0.0   0:00.03 ksoftirq
   12 root      20   0      0      0      0 S   0.0   0.0   0:00.17 rcu_sched
   13 root      20   0      0      0      0 S   0.0   0.0   0:00.01 migration
   15 root      20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   17 root      20   0      0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
   18 root      20   0      0      0      0 S   0.0   0.0   0:00.00 netns
   19 root      20   0      0      0      0 S   0.0   0.0   0:00.02 kworker/
   21 root      20   0      0      0      0 S   0.0   0.0   0:00.01 kauditd/
  297 root      20   0      0      0      0 S   0.0   0.0   0:00.00 khungtask
  298 root      20   0      0      0      0 S   0.0   0.0   0:00.00 oom_reaper
  299 root      20   0      0      0      0 S   0.0   0.0   0:00.00 writeback
  301 root      20   0      0      0      0 S   0.0   0.0   0:00.05 kcompactd0
  302 root      25   5      0      0      0 S   0.0   0.0   0:00.00 ksm
  303 root      39  19      0      0      0 S   0.0   0.0   0:00.00 khugepaged
  358 root      20   0      0      0      0 S   0.0   0.0   0:00.00 kintegrityd
  360 root      20   0      0      0      0 S   0.0   0.0   0:00.00 kblockd
  361 root      20   0      0      0      0 S   0.0   0.0   0:00.00 blkcg_punt_bio
  713 root      20   0      0      0      0 S   0.0   0.0   0:00.00 xen-balloon
  719 root      20   0      0      0      0 S   0.0   0.0   0:00.00 tpm_dev_wq
  725 root      20   0      0      0      0 S   0.0   0.0   0:00.00 md
  728 root      20   0      0      0      0 S   0.0   0.0   0:00.00 edac-poller
  733 root      20   0      0      0      0 S   0.0   0.0   0:00.00 watchdogd
```



### 3.3.2 Thread

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan Start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1.

- Tuliskan kode berikut ini dengan text editor

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>

void * thread1() {
    while(1) {
        printf("Hello!!\n");
    }
}
```

```

    }
}

void * thread2() {
    while(1) {
        printf("How are you?\n");
    }
}

int main() {
    int status;
    pthread_t tid1,tid2;
    pthread_create(&tid1,NULL,thread1,NULL);
    pthread_create(&tid2,NULL,thread2,NULL);
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
    return 0;
}

```

GNU nano 2.9.8 praktikum.txt Modified

```

#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>

void *thread1()
{
    while (1)
    {
        printf("Hello!!\n");
    }
}

void *thread2()
{
    while (1)
    {
        printf("How are you?\n");
    }
}

int main()
{
    int status;
    pthread_t tid1, tid2;
    pthread_create(&tid1, NULL, thread1, NULL);
    pthread_create(&tid2, NULL, thread2, NULL);
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    return 0;
}

```

config Dani Adrian.1 x + - □ ×

File Edit View ⚙

Dani Adrian  
2251502011111009

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

<sup>^</sup>G Get Help  
 <sup>^</sup>O Write Out  
 <sup>^</sup>W Where Is  
 <sup>^</sup>K Cut Text  
 <sup>^</sup>J Justify  
 <sup>^</sup>C Cur Pos  
 <sup>^</sup>U Undo  
 <sup>^</sup>A Mark Text  
 <sup>^</sup>] To Bracket  
<sup>^</sup>X Exit  
 <sup>^</sup>R Read File  
 <sup>^</sup>N Replace  
 <sup>^</sup>U Uncut Text  
 <sup>^</sup>T To Spell  
 <sup>^</sup>G Go To Line  
 <sup>^</sup>E Redo  
 <sup>^</sup>G Copy Text  
 <sup>^</sup>W WhereIs Next

b. Simpan berkas tersebut dengan nama **threadsatu.c**



e. Tuliskan kode berikut ini

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>

pthread_t tid[2];

void* doSomething(void *arg) {
    unsigned long i = 0;
    pthread_t id = pthread_self();

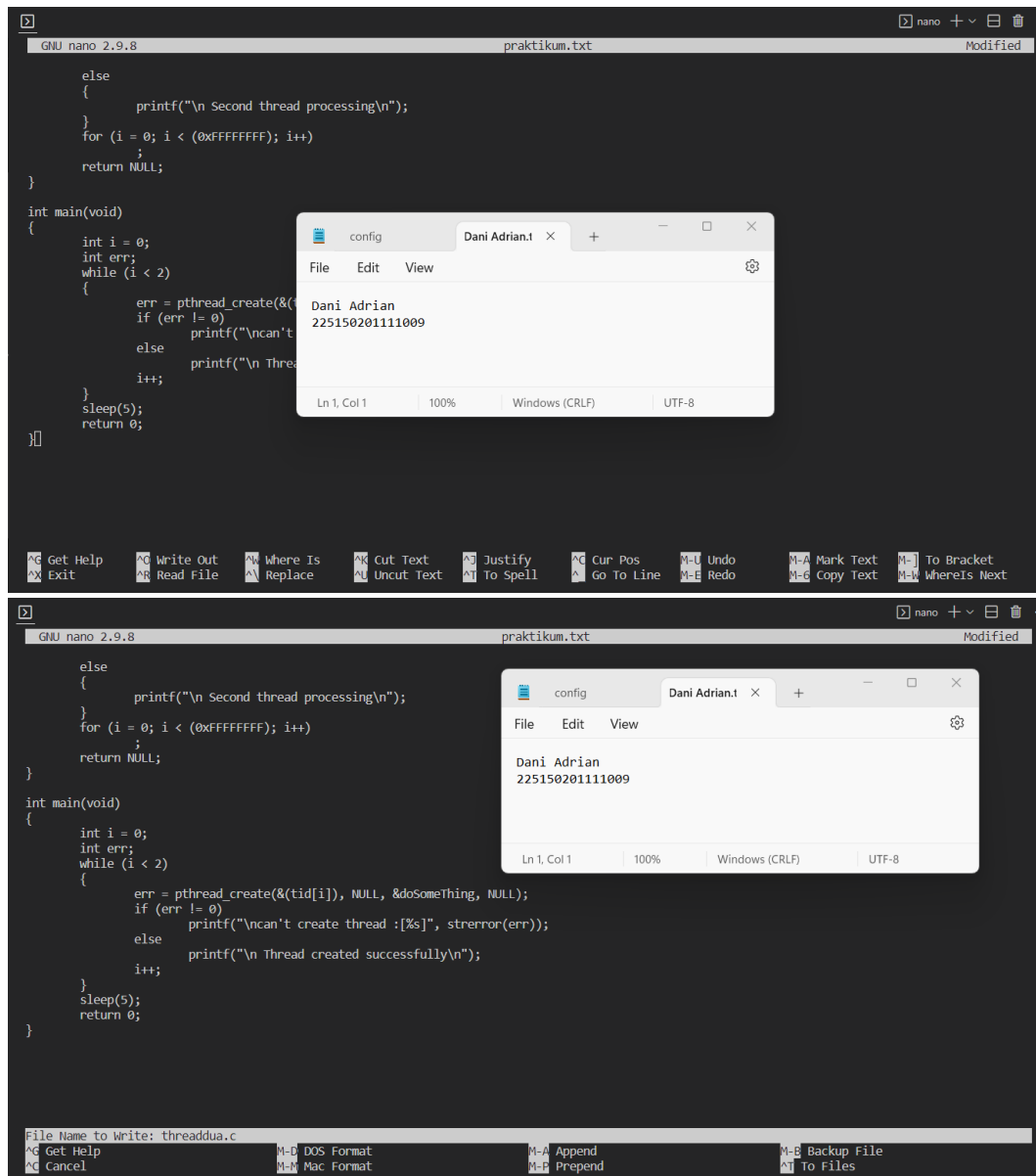
    if(pthread_equal(id,tid[0])) {
        printf("\n First thread processing\n");
    } else {
        printf("\n Second thread processing\n");
    }

    for(i=0; i<(0xFFFFFFFF);i++);
    return NULL;
}

int main(void) {
    int i = 0;
    int err;

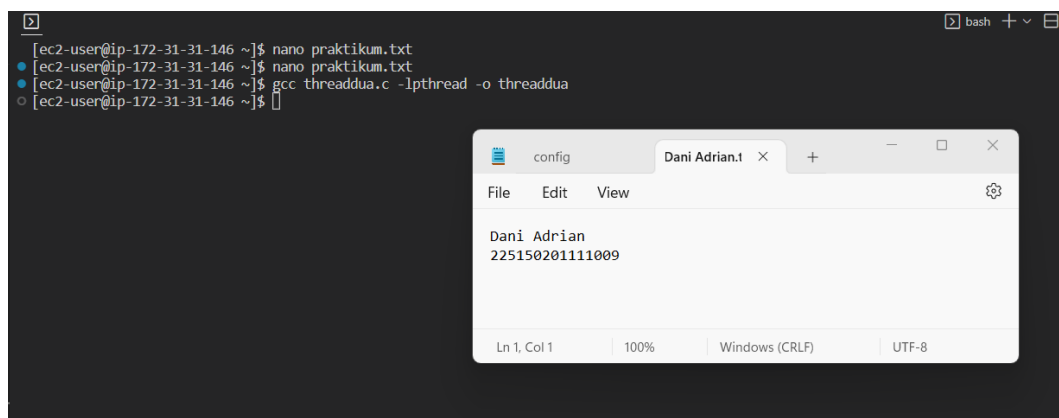
    while(i < 2) {
        err = pthread_create(&(tid[i]), NULL,
&doSomething, NULL);
        if (err != 0)
            printf("\ncan't create thread :[%s]",
strerror(err));
        else
            printf("\n          Thread          created
successfully\n");
        i++;
    }
    sleep(5);
    return 0;
}
```

f. Simpan berkas kedua ini dengan nama **threaddua.c**



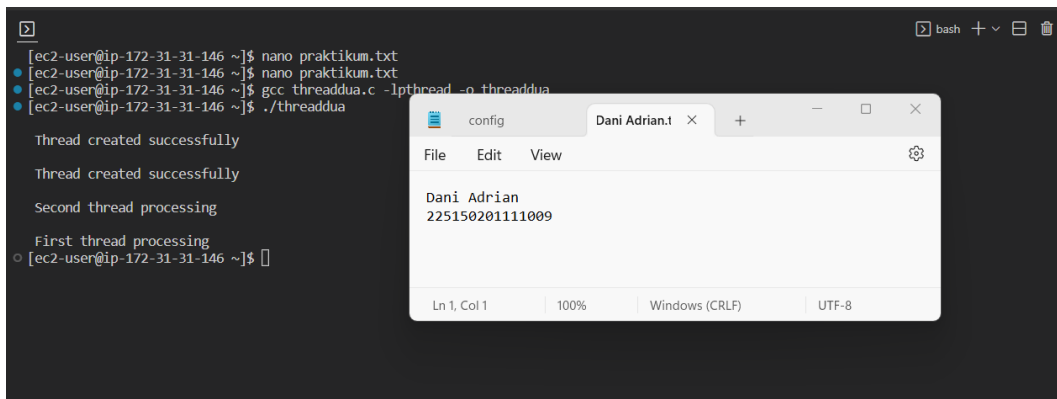
g. Kompilasi kode program **threaddua.c** melalui terminal dengan menuliskan perintah

```
[admin@host]$ gcc threaddua.c -lpthread -o threaddua
```





- h. Jalankan program baru tersebut diatas. Tunjukkan tampilan yang ada pada terminal serta berikan penjelasan singkat dari tampilan tersebut.



```
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ gcc threadua.c -lpthread -o threadua
[ec2-user@ip-172-31-31-146 ~]$ ./threadua

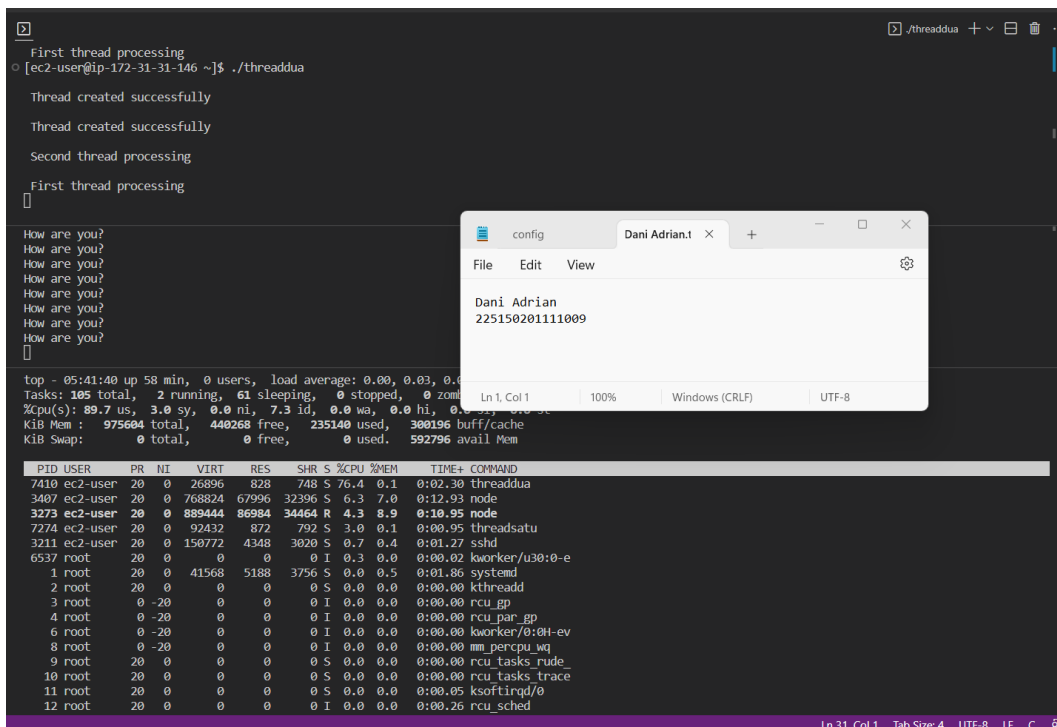
Thread created successfully

Thread created successfully

Second thread processing

First thread processing
[ec2-user@ip-172-31-31-146 ~]$
```

- i. Buka aplikasi terminal yang lain, dan jalankan kedua thread tersebut.
- j. Temukan *identitas proses* (dan mungkin juga thread) yang terkait dengan perintah eksekusi kedua thread tersebut!



```
First thread processing
[ec2-user@ip-172-31-31-146 ~]$ ./threadua

Thread created successfully

Thread created successfully

Second thread processing

First thread processing
[ec2-user@ip-172-31-31-146 ~]$

How are you?
How are you?
How are you?
How are you?
How are you?
How are you?
How are you?
How are you?
How are you?
[ec2-user@ip-172-31-31-146 ~]$

top - 05:41:40 up 58 min, 0 users, load average: 0.00, 0.03, 0.00
Tasks: 105 total, 2 running, 61 sleeping, 0 stopped, 0 zombie
%Cpu(s): 89.7 us, 3.0 sy, 0.0 ni, 7.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 975604 total, 448268 free, 235140 used, 300196 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 592796 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 7410 ec2-user   20   0 26896    828    748 S   0.0   0.1   0:02.30 threadua
 3407 ec2-user   20   0 768824   6796   3236 S   6.3   7.0   0:12.03 node
 3273 ec2-user   20   0 889444   86984  34464 R   4.3   8.9   0:10.95 node
 7274 ec2-user   20   0 92432    872    792 S   3.0   0.1   0:00.95 threadsatu
 7274 ec2-user   20   0 150772   4348   3020 S   0.7   0.4   0:01.27 sshd
 6537 root       20   0 0         0        0 I   0.3   0.0   0:00.02 kworker/u30:0-e
 1 root      20   0 41568    5188   3756 S   0.0   0.5   0:01.86 systemd
 2 root      20   0 0         0        0 S   0.0   0.0   0:00.00 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
 6 root      0 -20 0         0        0 I   0.0   0.0   0:00.00 kworker/0:0H-ev
 8 root      0 -20 0         0        0 I   0.0   0.0   0:00.00 mm_percpu_wq
 9 root      20   0 0         0        0 S   0.0   0.0   0:00.00 rcu_tasks_rude
10 root      20   0 0         0        0 S   0.0   0.0   0:00.00 rcu_tasks_trace
11 root      20   0 0         0        0 S   0.0   0.0   0:00.05 ksoftirqd/0
12 root      20   0 0         0        0 I   0.0   0.0   0:00.26 rcu_sched
```

- k. Apakah yang bisa dijelaskan dari kedua contoh thread diatas? Apa yang menjadi masalah utama dalam hal ini?

Pada threadsatu, pthread\_create merupakan fungsi yang digunakan untuk menciptakan thread baru dalam suatu proses dengan atribut yang ditentukan. Bila tidak diisi, maka akan diisi dengan atribut default. Fungsi ini mempunyai 4 parameter yakni pthread\_create(&thread1, NULL, function, NULL).

Pthread\_join merupakan fungsi untuk melakukan penggabungan dengan thread lain yang telah diterminasi (exit). Namun bila thread tersebut belum di terminasi maka fungsi ini akan menunggu sampai thread tersebut terminated.

Terjadinya infinite looping karena kemungkinan besar tidak ada batasan sampai kapan looping akan berhenti. Ketika thread pertama selesai, namun karena tidak ada bayasan, maka thread kembali dieksekusi bersamaan dengan pengeksekusian thread kedua, menyebabkan proses terus berjalan tanpa berhenti kecuali user menghentikan secara paksa sedangkan pada threaddua didefinisikan batasannya sebanyak dua kali yaitu  $I < 2$  (I dimulai dari 0), sehingga tidak terjadi infinite looping. Setelah memproses dua thread, terdapat fungsi sleep (5) untuk tertidur atau menunggu thread yang sedang berjalan dalam jangka waktu tertentu dan pada kasus diatas itu terjadi selama 5 detik

1. Berikut ini adalah sebuah kode program yang dijalankan secara sekuensial. Dibentuk menjadi satu proses dengan *single thread*.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static int num_accts = 1024; // number of bank
accounts
static int num_trans = 10000; // number of
transactions
static int think_time = 50; // amount of "thinking
time"

struct acct_t { int bal; };

/*
 * Pointer to accounts
 */
struct acct_t *accts;

/*
 * Method to perform a number of transactions.
 * Parameter "dummy" is not used.
 */
void *transact(void *dummy) {
```

```

int i;
for (i = 0; i < num_trans; i++) {
    // pick two random accounts
    int acct_id_from = rand() % num_accts;
    int acct_id_to = rand() % num_accts;

    // pick a random amount
    int amt = rand() % 100;

    // try to transfer the money
    if (accts[acct_id_from].bal > amt) {
        accts[acct_id_from].bal -= amt;
        accts[acct_id_to].bal += amt;

        // "thinking time"... don't modify
this code!!
        amt *= think_time; while (amt--);
    }
}

int main(int argc, char **argv) {
    // make sure the number of arguments is odd
    (including the program name)
    if (!(argc == 1 || argc == 3 || argc == 5 ||
argc == 7)) {
        fprintf(stderr, "usage: %s [-a <accts>]
[-i <transactions>] [-t <think-time>]\n",
            argv[0]);
        exit(-1);
    }

    // look at each runtime argument and see
    which value it's attempting to set
    int i;
    for (i = 1; i < argc; i++) {
        if (!strcmp(argv[i], "-a")) {
            num_accts = atoi(argv[i+1]);
            i++;
        } else if (!strcmp(argv[i], "-i")) {
            num_trans = atoi(argv[i+1]);
            i++;
        }
    }
}

```

```

        } else if (!strcmp(argv[i], "-t")) {
            think_time = atoi(argv[i+1]);
            i++;
        } else {
            fprintf(stderr, "usage:  %s  [-a
<accts>] [-i <transactions>] [-t <thinktime>]\n",
argv[0]);
            exit(-1);
        }
    }

    // display the parameters that will be used
for this test run
    fprintf(stderr, "%s: -a %d -i %d -t %d\n",
argv[0], num_accts, num_trans, think_time);

    // initialize the random number generator
    srand(1);

    // create the bank accounts
    accts = (struct acct_t *)malloc(num_accts *
sizeof(struct acct_t));

    // initialize the bank accounts' values and
keep track of the total sum in all accounts
    int original_sum = 0;
    for (i = 0; i < num_accts; i++) {
        accts[i].bal = rand() % 1000;
        original_sum += accts[i].bal;
    }

    // call the transact function to do the
transfers
    transact(NULL);

    // find the total sum of all accounts after
the transfers are done
    int sum = 0;
    for (i = 0; i < num_accts; i++) {
        sum += accts[i].bal;
    }

```

```

        // if the sum is not equal to the original
        sum, then we had a race condition!!
        if (sum != original_sum) {
            fprintf(stderr, "ERROR!  original_sum
= %d, sum = %d\n", original_sum, sum);
        } else {
            fprintf(stderr, "Values  are  still
consistent\n");
        }
        return 0;
    }
}

```

```

GNU nano 2.9.8      praktikum.txt      Modified
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static int num_accts = 1024; // number of bank accounts
static int num_trans = 10000; // number of transactions
static int think_time = 50; // amount of "thinking time"

struct acct_t
{
    int bal;
};

/*
 * Pointer to accounts
 */
struct acct_t *accts;

/*
 * Method to perform a number of transactions.
 * Parameter "dummy" is not used.
 */
void *transact(void *dummy)
{
    int i;
    for (i = 0; i < num_trans; i++)
    {
        // pick two random accounts
        int acct_id_from = rand() % num_accts;
        int acct_id_to = rand() % num_accts;
        // pick a random amount
    }
}

```

m. Kompilasi kode program tersebut dan simpan dengan nama berkas **singlethread**.

```

GNU nano 2.9.8      praktikum.txt      Modified
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static int num_accts = 1024; // number of bank accounts
static int num_trans = 10000; // number of transactions
static int think_time = 50; // amount of "thinking time"

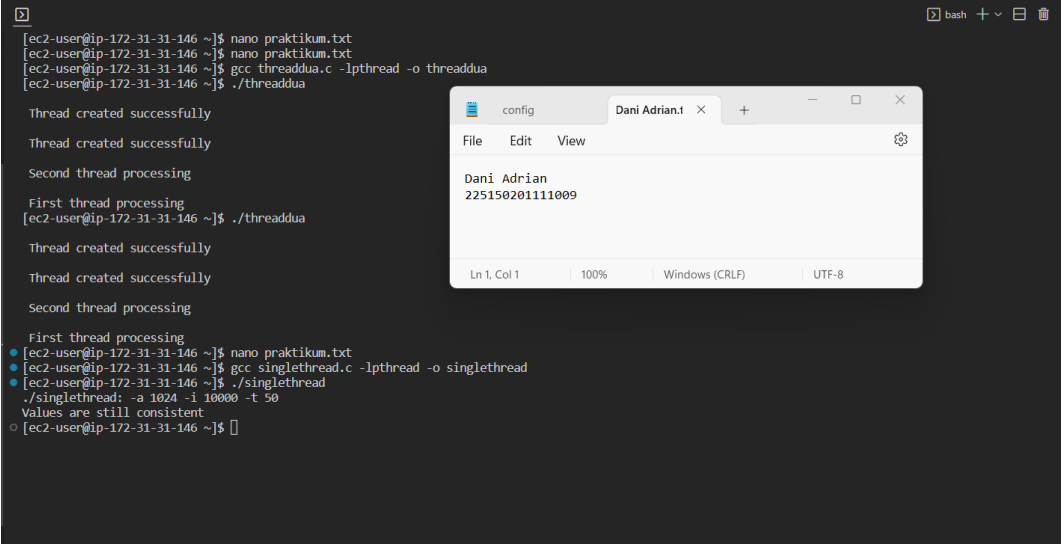
struct acct_t
{
    int bal;
};

/*
 * Pointer to accounts
 */
struct acct_t *accts;

/*
 * Method to perform a number of transactions.
 * Parameter "dummy" is not used.
 */
void *transact(void *dummy)
{
    int i;
    for (i = 0; i < num_trans; i++)
    {
        // pick two random accounts
        int acct_id_from = rand() % num_accts;
        int acct_id_to = rand() % num_accts;
        // pick a random amount
    }
}

```

- n. Jalankan, tunjukkan dan berikan penjelasan singkat terkait eksekusi program tersebut.

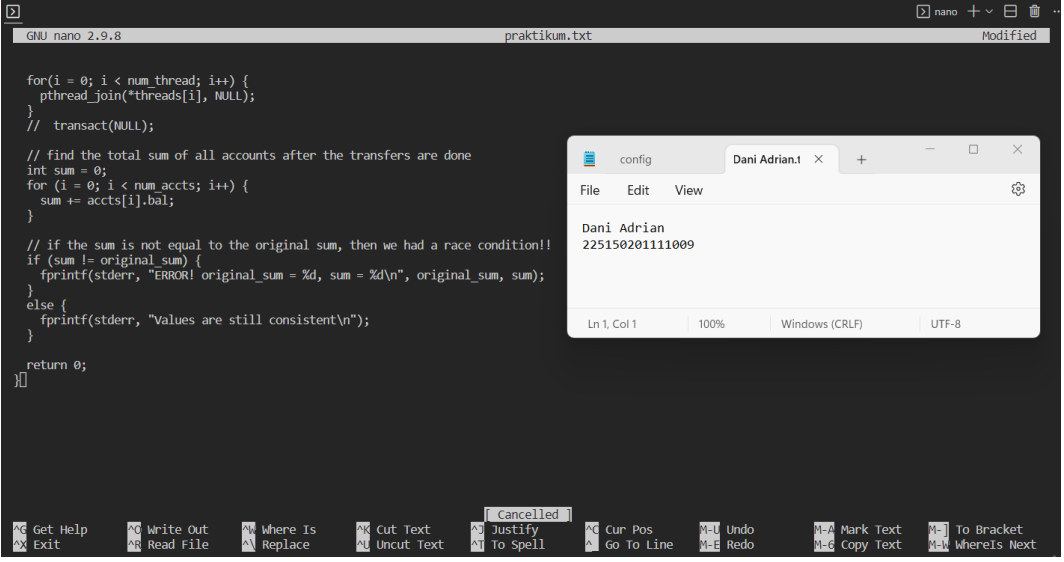


```
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ gcc threaddua.c -lpthread -o threaddua
[ec2-user@ip-172-31-31-146 ~]$ ./threaddua

Thread created successfully
Thread created successfully
Second thread processing
First thread processing
[ec2-user@ip-172-31-31-146 ~]$ ./threaddua
Thread created successfully
Thread created successfully
Second thread processing
First thread processing
[ec2-user@ip-172-31-31-146 ~]$ nano praktikum.txt
[ec2-user@ip-172-31-31-146 ~]$ gcc singlethread.c -lpthread -o singlethread
[ec2-user@ip-172-31-31-146 ~]$ ./singlethread
./singlethread: -a 1024 -i 10000 -t 50
Values are still consistent
[ec2-user@ip-172-31-31-146 ~]$
```

Output tersebut digunakan untuk mengecek perubahan pada thread. Apabila tidak ada perubahan maka akan muncul output “values are still consistent”

- o. Modifikasi kode program tersebut menjadi beberapa thread yang berbeda dan simpan dengan nama berkas **threadtiga**.



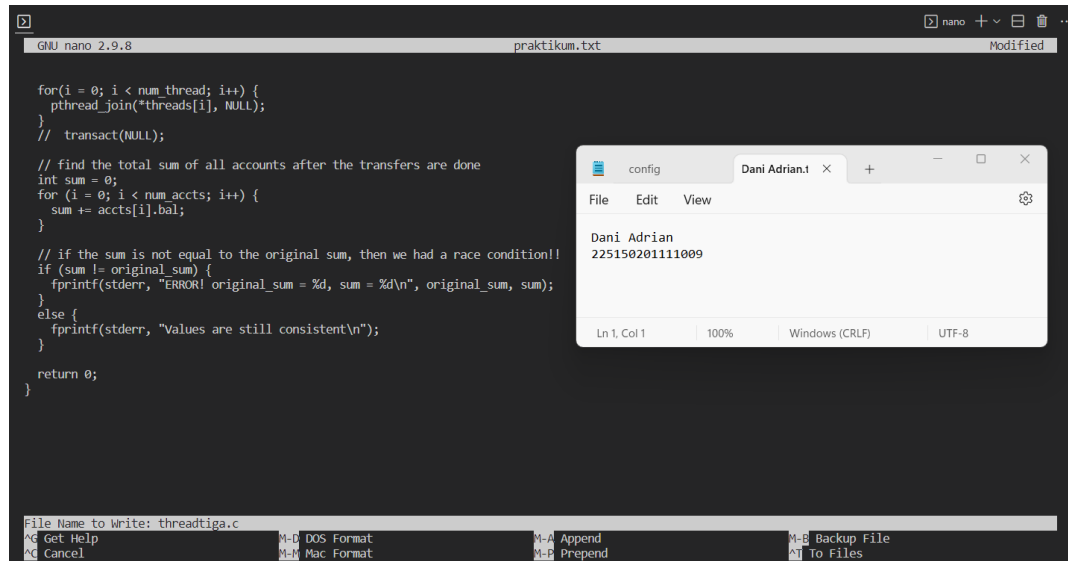
```
GNU nano 2.9.8      praktikum.txt      Modified

for(i = 0; i < num thread; i++) {
    pthread_join(*threads[i], NULL);
}
// transact(NULL);

// find the total sum of all accounts after the transfers are done
int sum = 0;
for (i = 0; i < num accts; i++) {
    sum += accts[i].bal;
}

// if the sum is not equal to the original sum, then we had a race condition!!
if (sum != original_sum) {
    fprintf(stderr, "ERROR! original_sum = %d, sum = %d\n", original_sum, sum);
}
else {
    fprintf(stderr, "Values are still consistent\n");
}

return 0;
}
```



```
GNU nano 2.9.8      praktikum.txt      Modified

for(i = 0; i < num_thread; i++) {
    pthread_join(*threads[i], NULL);
}
// transact(NULL);

// find the total sum of all accounts after the transfers are done
int sum = 0;
for (i = 0; i < num_accts; i++) {
    sum += accts[i].bal;
}

// if the sum is not equal to the original sum, then we had a race condition!!
if (sum != original_sum) {
    fprintf(stderr, "ERROR! original_sum = %d, sum = %d\n", original_sum, sum);
}
else {
    fprintf(stderr, "Values are still consistent\n");
}

return 0;
}
```

File Name to Write: threadtiga.c

⌘ Get Help ⌘ M-D DOS Format ⌘ M-A Append ⌘ M-B Backup File  
⌘ Cancel ⌘ M-M Mac Format ⌘ M-P Prepend ⌘ M-T To Files

p. Deskripsikan hasil eksekusi program **threadtiga** beserta penjelasannya.

Setelah kode program singlethread.c dimodifikasi dan disimpan menjadi threadtiga.c kemudian dieksekusikan dengan ./threadtiga menghasilkan output yang tidak jauh berbeda, masih mencetak "Values are still consistent" namun terdapat tambahan keluaran -p yang merupakan banyaknya thread dan kejelasan thread yang sedang berjalan dengan "Join thread: " hasil dari fungsi pthread\_join

Nilai masih konsisten karena tidak ada nilai yang berubah ketika proses dijalankan.



## **Tugas**

### **3.4.1 Proses**

1. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 1. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

\$man ps

Manfaat dari perintah "man ps" adalah untuk menampilkan manual atau deskripsi tentang perintah "ps" yang terdapat pada command line. Ketika menjalankan "man ps", akan ditampilkan berbagai informasi tentang perintah tersebut, seperti nama, deskripsi, dan contoh penggunaannya. Perintah "ps" sendiri berguna untuk menampilkan daftar proses yang sedang berjalan pada sistem, dan dapat digunakan dengan opsi tertentu untuk menampilkan informasi yang lebih spesifik tentang proses-proses tersebut. Beberapa opsi yang umum digunakan pada perintah "ps" antara lain "-a", "-u", "-x", "-e", dan "-f". Sebagai contoh, penggunaan perintah "ps -ef" akan menampilkan informasi detail tentang semua proses yang sedang berjalan, seperti username pemilik proses, ID proses, dan waktu yang telah dijalankan.

2. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 2. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

\$top

Perintah "top" pada sistem Linux berfungsi untuk menampilkan daftar proses yang sedang berlangsung secara real-time tanpa menggunakan antarmuka grafis (GUI). Dengan menggunakan perintah "top", pengguna dapat melakukan manajemen proses pada sistem secara efektif. Ketika perintah "top" dijalankan, akan muncul daftar semua proses yang sedang berjalan pada sistem, dan informasi ini akan diperbarui setiap detik agar pengguna dapat melihat aktivitas sistem secara real-time.

3. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 3. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**



\$ps

Perintah "ps" pada sistem Linux berfungsi untuk menampilkan daftar proses yang sedang berjalan, di mana setiap proses akan memiliki Process ID (PID), nama terminal pengendali (TTY), waktu CPU kumulatif dalam menit dan detik (TIME), serta nama perintah yang digunakan untuk memulai proses (CMD). PID merupakan informasi penting bagi pengguna untuk menghentikan proses yang tidak lagi digunakan. TTY menyimpan nama terminal pengendali yang digunakan oleh proses. Waktu CPU kumulatif ditampilkan dalam format menit dan detik dan menunjukkan berapa lama CPU telah digunakan oleh suatu proses. CMD berisi nama perintah yang digunakan untuk memulai proses. Dengan menggunakan perintah "ps", pengguna dapat memonitor dan mengelola proses yang sedang berjalan pada sistem Linux.

4. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 4. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

\$ps -ax

Perintah "ps -ax" pada sistem Linux merupakan gabungan dari tiga perintah, yaitu "ps", "a", dan "x". Perintah "ps" digunakan untuk menampilkan daftar proses yang sedang berjalan pada sistem. Perintah "a" menampilkan semua proses, termasuk yang dimulai oleh pengguna dan sistem. Sedangkan perintah "x" memasukkan proses yang tidak terhubung dengan terminal. Dengan menggunakan gabungan dari ketiga perintah tersebut, pengguna dapat menampilkan daftar semua proses yang sedang berjalan pada sistem, termasuk yang tidak terhubung dengan terminal. Hal ini sangat berguna untuk memonitor dan mengelola proses pada sistem Linux secara lebih efektif.

5. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 7 (a-c). Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Dari hasil eksekusi perintah "ps -ax" yang menampilkan daftar proses yang sedang berjalan pada sistem, terlihat bahwa proses dengan PID atau Process ID 5496 sedang menjalankan perintah untuk membuka file "Praktikum.txt" menggunakan aplikasi nano.

6. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 8. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Saat perintah "kill -9 5496" dieksekusi di terminal pertama, terlihat keluaran "Killed" pada terminal kedua yang sedang membuka file "Praktikum.txt" menggunakan aplikasi nano. Hal ini menunjukkan bahwa proses nano Praktikum.txt telah berhasil dimatikan dengan menggunakan PID 5496 yang diidentifikasi sebelumnya. Dengan mengetahui PID dari proses yang sedang berjalan, pengguna dapat menghentikan atau mengontrol proses tersebut secara efektif pada sistem Linux.

7. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 9. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Di terminal kedua dieksekusi perintah "pstree", yang berfungsi untuk menampilkan daftar proses yang sedang berjalan pada sistem Linux dalam bentuk pohon. Dengan menggunakan perintah ini, pengguna dapat melihat hierarki proses dan hubungan antara satu proses dengan proses lainnya pada sistem operasi Linux.

8. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 10. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

\$ sudo yum update

Perintah APT merupakan kepanjangan dari Advanced Package Tool, dan berfungsi untuk memperbarui daftar paket yang tersedia pada sistem operasi Linux. APT merupakan sebuah database yang menyimpan daftar paket yang tersedia dari repository yang telah diaktifkan di sistem.

Terjadinya error permission denied di awal kemungkinan disebabkan oleh akses yang tidak mencukupi untuk membuka file pada ec2 AWS. Untuk mengatasi hal ini, pengguna perlu menggunakan perintah "sudo", singkatan dari "Super User Do". Perintah ini memungkinkan pengguna untuk menjalankan perintah sebagai superuser, meskipun menggunakan akun pengguna biasa, dengan mengikuti kebijakan keamanan yang telah ditentukan.

9. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 11. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

\$pstree

Pada terminal kedua, perintah \$pstree dijalankan untuk menampilkan daftar proses yang sedang berjalan pada sistem Linux dalam bentuk struktur pohon.

10. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 12. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Ketika sedang menjalankan perintah sudo yum update di terminal kedua, di terminal pertama terlihat informasi tentang proses tersebut pada kolom Command di Top. Hal ini menandakan bahwa proses tersebut sedang berjalan pada sistem.

Perintah control + C digunakan untuk menghentikan proses yang sedang berjalan secara paksa. Perintah ini biasanya digunakan ketika proses terjebak dalam looping yang tidak terbatas atau tidak merespon perintah pengguna. Dengan menggunakan perintah control + C, proses yang sedang berjalan dapat dihentikan dengan cepat dan aman.

11. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 13. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Perintah ini digunakan untuk menampilkan tampilan proses yang sedang berjalan pada sistem Linux dalam bentuk pohon.

12. Apakah hasil yang anda dapatkan setelah menjalankan perintah pada Langkah 1-7 pada bagian 2.42. Jelaskan apa yang anda ketahui mengenai perintah ini dan bagaimana informasi yang didapatkan.

**Jawab:**

Nano adalah sebuah text editor pada sistem operasi Linux. Untuk menyimpan kode program dengan nama fork.c, kita dapat menggunakan nano. Untuk melakukan kompilasi pada fork.c, kita dapat menggunakan gcc. Jika gcc belum terinstal, kita dapat menggunakan perintah "sudo apt install gcc" untuk menginstallnya. Setelah itu, gunakan perintah "gcc nama\_file"

atau "gcc fork.c" untuk mengcompile kode program dan membuat file output executable dengan nama a.out. Jika tidak ada nama output yang ditentukan, maka nama file output executable akan menjadi a.out. Untuk mengeksekusi kode program fork.c, kita dapat menggunakan perintah "./a.out". Kemudian, untuk melihat child dari parent process yang sedang dijalankan, kita dapat menggunakan perintah "pstree". Sedangkan untuk menampilkan penggunaan resources seperti processor dan memori dari proses yang sedang berjalan, kita dapat menggunakan perintah "top".

13. Apa hubungan antara fork() dan exec() dengan proses yang berlangsung pada sistem operasi?

**Jawab:**

Perintah fork() digunakan untuk membuat duplikat proses yang sama persis dengan parent-nya. Sementara perintah exec() digunakan untuk mengganti seluruh program dengan program baru.

Dalam penggunaannya, fork() dan exec() biasanya digunakan bersama-sama untuk membuat proses baru. Dengan fork(), proses parent dan child akan memiliki salinan yang sama persis, dan keduanya akan berjalan secara paralel. Sedangkan dengan exec(), program baru dapat dijalankan di dalam child process yang telah dibuat oleh fork().

14. Bagaimana fork() dan exec() mempengaruhi struktur direktori, penggunaan memori dan CPU?

**Jawab:**

Saat menjalankan fork(), akan terjadi duplikasi proses yang menghasilkan 2 proses yang identik yaitu parent dan child. Kedua proses ini berjalan secara bersamaan dan saling berbagi program dan data pada memori. Sedangkan pada saat menjalankan exec(), proses child yang dihasilkan akan berbeda dengan parent karena program yang baru dimuat ke dalam alamat dengan kode dan data segmen baru.

Ketika fork() dieksekusi, setiap proses akan memiliki PCB (Process Control Block) masing-masing, termasuk counter, register, dan PID. PCB ini digunakan untuk mengontrol dan memantau proses tersebut. Namun, ketika exec() dieksekusi, proses child dimuat pada alamat yang sama dengan proses parent dan PCB tidak berubah karena exec() hanya mengganti program pada proses child tanpa mempengaruhi proses parent.

### 3.4.2 Thread

15. Dari langkah (a) sampai Langkah (n), jelaskan apa yang anda ketahui tentang:
- (a) thread
  - (b) single thread
  - (c) multi thread

**Jawab:**

Thread adalah unit terkecil dari sebuah proses yang bisa dieksekusi secara independen. Proses dapat memiliki beberapa thread yang dapat bekerja secara asinkron untuk menangani tugas yang berbeda-beda. Pada system operasi modern, thread menjadi sangat penting karena meningkatkan efisiensi dan performa dari sebuah proses.

Single Thread adalah proses yang hanya memiliki satu thread yang dieksekusi. Hal ini membuat proses menjadi kurang efisien karena sumber daya proses hanya digunakan untuk satu tugas yang dilakukan oleh satu thread saja. Contohnya adalah praktikum menggunakan singlethread.c yang menunjukkan bahwa saat ada transaksi bank yang dilakukan secara bersamaan, hal ini dapat menimbulkan masalah data race yang membuat variabel tidak valid.

Pada praktikum tersebut, hasil output menunjukkan bahwa nilai total saldo akun nasabah tetap konsisten, menandakan bahwa tidak ada bagian dari kode yang memproses variabel global yang telah didefinisikan.

Multi Thread adalah proses dengan banyak thread yang mengerjakan lebih dari satu tugas dalam satu waktu. Sistem ini lebih efisien karena thread dapat dieksekusi secara bersamaan secara paralel, meningkatkan utilitas CPU dan menghemat penggunaan memori.

16. Dari langkah (o) sampai langkah (p), apa yang anda ketahui dari hasil eksekusi program ini? Mengapa demikian dan apa yang menjadi dasar penjelasan anda?

**Jawab:**

Setelah melakukan modifikasi pada kode program singlethread.c dan menyimpannya menjadi threadtiga.c, kemudian dieksekusi dengan perintah /threadtiga, output yang dihasilkan tidak terlalu berbeda dengan yang sebelumnya, yaitu masih menampilkan pesan "Values are still consistent". Namun, ada tambahan output "-p" yang menunjukkan jumlah thread dan informasi thread yang sedang berjalan, ditampilkan hasil dari fungsi pthread\_join yang dieksekusi.

Hal ini menunjukkan bahwa nilai tetap konsisten karena tidak ada perubahan nilai yang terjadi selama proses berjalan, dan informasi tambahan tersebut

memberikan pemahaman lebih tentang jumlah thread yang digunakan dan status thread yang sedang berjalan menggunakan fungsi `pthread_join`.



## **Kesimpulan**

Tuliskan kesimpulan yang dapat diperoleh dari hasil percobaan ini berdasarkan hasil pembahasan yang anda buat.

Proses adalah sebuah program atau aplikasi yang sedang berjalan pada sistem operasi. Setiap proses memiliki ruang memori dan sumber daya yang tersedia seperti CPU, memori, I/O, dan lain-lain. Dalam sebuah sistem operasi modern, proses dapat memiliki beberapa thread, dimana setiap thread masing-masing memiliki program dan data sendiri yang berjalan secara independen dalam proses yang sama. Dalam konteks ini, thread adalah sebuah unit terkecil dari eksekusi program yang dapat dijadwalkan oleh sistem operasi untuk dieksekusi secara bersamaan dengan thread lainnya dalam proses yang sama.

Perbedaan antara proses dan thread terletak pada sumber daya yang digunakan dan cara pengaturan penggunaannya. Setiap proses memiliki alokasi memori dan sumber daya yang terpisah, sedangkan setiap thread berbagi memori dan sumber daya yang sama dalam proses yang sama. Karena itu, thread dapat dijalankan lebih cepat dan lebih efisien daripada proses, karena overhead pembuatan proses lebih besar dibandingkan pembuatan thread. Namun, thread juga memiliki risiko lebih besar dalam menghasilkan kesalahan atau bug, karena setiap thread berbagi memori dan sumber daya yang sama, sehingga perubahan yang dilakukan oleh satu thread dapat mempengaruhi thread lainnya.