

# **Laporan Pratikum**

## **Sistem Operasi Dasar**

PRATIUM 8



Dosen Pengampu : Ibnu Surya, S.T.,M.T.  
AIL : Iqbal Mahatma Putra S.S.T

Nama : Laurin Madelau  
NIM : 2055301067  
Kelas : 1 TI C

**JURUSAN TEKNIK INFORMATIKA**  
**POLITEKNIK CALTEX RIAU**  
**2021**

## Percobaan 1: Status Proses

1. Instruksi ps (process status) digunakan untuk melihat kondisi proses yang ada. PID adalah Nomor Identitas Proses, TTY adalah nama terminal dimana proses tersebut aktif, STAT berisi S (Sleeping) dan R (Running), COMMAND merupakan instruksi yang digunakan.

\$ ps

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps
  PID TTY          TIME CMD
 2168 pts/0    00:00:00 bash
 2634 pts/0    00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

- ⇒ Perintah diatas digunakan untuk melihat kondisi proses. Dan ketika perintah ini dieksekusi maka informasi yang ditampilkan berupa:
- PID yang berfungsi untuk menampilkan Nomor Identitas Proses.
  - TTY menampilkan nama terminal dimana proses tersebut aktif.
  - TIME berfungsi menampilkan waktu yang diperlukan dalam mengakses perintah.
  - CMD (Command) yang berfungsi untuk menampilkan instruksi/perintah yang digunakan.

2. Untuk melihat faktor/elemen lainnya, gunakan option -u (user). %CPU adalah presentasi CPU time yang digunakan oleh proses tersebut, %MEM adalah presentasi system memori yang digunakan proses, SIZE adalah jumlah memori yang digunakan, RSS (Real System Storage) adalah jumlah memori yang digunakan, START adalah kapan proses tersebut diaktifkan.

\$ ps -u

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -u
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
laurinm+     1421  0.0  0.3 172632  3252 tty2      Ssl+  13:31   0:00 /usr/lib/gdm
laurinm+     1423  0.8  3.0 240324 30312 tty2      Sl+   13:31   0:02 /usr/lib/xor
laurinm+     1465  0.0  0.4 199608  4968 tty2      Sl+   13:31   0:00 /usr/libexec
laurinm+     2168  0.0  0.3  19252  3752 pts/0     Ss    13:33   0:00 bash
laurinm+     2649  0.0  0.3  20112  3112 pts/0     R+    13:37   0:00 ps -u
laurinmadelau@laurinmadelau-VirtualBox:~$
```

- ⇒ Perintah ini digunakan untuk melihat faktor/elemen user, yang kemudian dikombinasikan dengan menggunakan option-u dan ketika perintah ini dijalankan maka akan tampil data/informasi berupa:
- USER yang berfungsi memberikan informasi mengenai user yang sedang digunakan dalam proses tersebut.
  - PID yang berfungsi memberikan informasi mengenai nomor identitas dari proses yang ditunjukkan.
  - %CPU yang berfungsi untuk mempresentasikan waktu yang digunakan oleh CPU dalam proses tersebut.
  - %MEM berfungsi untuk mempresentasikan system memori yang digunakan dalam proses.
  - RSS (Real System Storage) berfungsi untuk memberikan informasi mengenai jumlah memori yang digunakan.
  - START berfungsi memberikan informasi mengenai kapan proses tersebut diaktifkan.

3. Mencari proses yang spesifik pemakai. Proses di atas hanya terbatas pada proses milik pemakai, dimana pemakai tersebut melakukan login.

\$ ps -u studentX

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -u laurinmadelau
```

PID	TTY	TIME	CMD
1363	?	00:00:00	systemd
1365	?	00:00:00	(sd-pam)
1372	?	00:00:00	pulseaudio
1374	?	00:00:00	tracker-miner-f
1377	?	00:00:00	gnome-keyring-d
1381	?	00:00:00	dbus-daemon
1385	?	00:00:00	gvfsd
1390	?	00:00:00	gvfsd-fuse
1408	?	00:00:00	gvfs-udisks2-vo
1415	?	00:00:00	gvfs-goa-volume

- ⇒ Perintah ini digunakan untuk melihat/mencari proses yang dijalankan oleh pengguna. Proses diatas hanya terbatas pada proses yang dijalankan oleh pengguna, dimana pemakai/pengguna tersebut melakukan login.

4. Mencari proses lainnya gunakan option a (all) dan au (all user).

\$ ps -a

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -a
  PID TTY          TIME CMD
 1423 tty2      00:00:04 Xorg
 1465 tty2      00:00:00 gnome-session-b
 2669 pts/0      00:00:00 ps
```

⇒ Perintah diatas digunakan untuk mengeksekusi perintah pada satu user saja.

\$ ps -au

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -au
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
laurinm+     1421  0.0  0.3 172632  3252 tty2      Ssl+  13:31   0:00 /usr/lib/gdm
laurinm+     1423  0.9  3.1 240324 31332 tty2      Sl+   13:31   0:04 /usr/lib/xor
laurinm+     1465  0.0  0.4 199608  4968 tty2      Sl+   13:31   0:00 /usr/libexec
laurinm+     2168  0.0  0.3  19252  3792 pts/0     Ss    13:33   0:00 bash
laurinm+     2670  0.0  0.3  20112  3184 pts/0     R+    13:39   0:00 ps -au
laurinmadelau@laurinmadelau-VirtualBox:~$
```

⇒ Perintah diatas digunakan untuk melihat informasi dari proses yang dijalankan oleh semua user.

## Percobaan 2: Sinyal

1. Membuat shell script dengan nama loop.sh

\$ vi loop.sh

## Sebuah shell script : loop.sh

while [ 1 ]

do

echo ".\c"

sleep 10

done

```
## Sebuah shell script : loop.sh
while [ 1 ]
do
echo ".\c"
sleep 10
done
~
```

laurinmadelau@laurinmadelau-VirtualBox:~\$ vi loop.sh

laurinmadelau@laurinmadelau-VirtualBox:~\$

2. Eksekusi file loop.sh sebagai background

```
$ chmod +x loop.sh
```

```
$ ./loop.sh &
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ chmod +x loop.sh
laurinmadelau@laurinmadelau-VirtualBox:~$ ./loop.sh &
[1] 3175
laurinmadelau@laurinmadelau-VirtualBox:~$
```

3. Melihat proses id

```
$ ps
```

```

  PID TTY          TIME CMD
 3068 pts/0    00:00:00 bash
 3175 pts/0    00:00:00 bash
 3186 pts/0    00:00:00 sleep
 3189 pts/0    00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

4. Menghentikan proses. Nomor 15 (SIGTERM) merupakan default

```
$ kill -15 [nomor PID] atau
```

```
$ kill [nomor PID]
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ .\c
.\c
kill -15 3175
[1]+  Terminated                  ./loop.sh
laurinmadelau@laurinmadelau-VirtualBox:~$
```

5. Menghentikan proses secara mutlak

```
$ kill -9 [nomor PID]
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ kill -9 3175
bash: kill: (3175) - No such process
laurinmadelau@laurinmadelau-VirtualBox:~$
```

### Percobaan 3: Mengelola sinyal

#### 1. Membuat file prog.sh

```
$ vi prog.sh
```

```
#!/bin/sh
```

```
echo "Program berjalan ..."
```

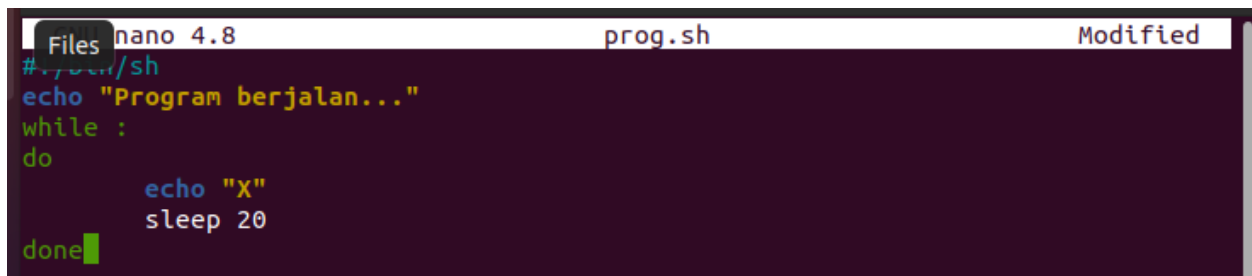
```
while :
```

```
do
```

```
echo "X"
```

```
sleep 20
```

```
done
```



```
Files nano 4.8 prog.sh Modified
#!/bin/sh
echo "Program berjalan..."
while :
do
    echo "X"
    sleep 20
done
```

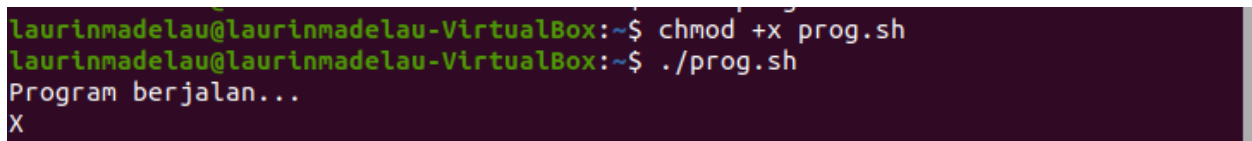
⇒ Membuat sebuah program pada file prog.sh

#### 2. Jalankan program tersebut. Karena program melakukan

looping, maka stop dengan mengirim sinyal interrupt (^C)

```
$ chmod +x prog.sh
```

```
$ ./prog.sh
```



```
laurinmadelau@laurinmadelau-VirtualBox:~$ chmod +x prog.sh
laurinmadelau@laurinmadelau-VirtualBox:~$ ./prog.sh
Program berjalan...
X
```

⇒ Pada perintah `chmod +x prog.sh` digunakan untuk memberi izin execute pada file prog.sh.  
`./prog.sh` digunakan untuk menjalankan program pada file prog.sh

3. Jalankan program tersebut sebagai background. Catat nomor PID proses, tekan Enter untuk ke foreground dan periksa melalui instruksi ps

```
$ ./prog.sh &
```

```
$ ps
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ./prog.sh &
[1] 3420
laurinmadelau@laurinmadelau-VirtualBox:~$ Program berjalan...
X
ps
  PID TTY          TIME CMD
  3068 pts/0        00:00:00 bash
  3420 pts/0        00:00:00 prog.sh
  3421 pts/0        00:00:00 sleep
  3422 pts/0        00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

⇒ Program prog.sh dijalankan ke proses background. Perintah ps digunakan untuk melihat proses-proses yang berjalan

4. Kirimkan sinyal terminasi sebagai berikut

```
$ kill [Nomor PID]
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ X
X
X
kill 3420
```

⇒ Pada perintah diatas digunakan untuk memberhentikan program.

5. Ubahlah program prog.sh dengan instruksi trap untuk menangkap sinyal yang dikirim

```
$ vi prog.sh
```

```
#!/bin/sh
```

```
trap "" 1 2 3 15
```

```
echo "Program berjalan ..."
```

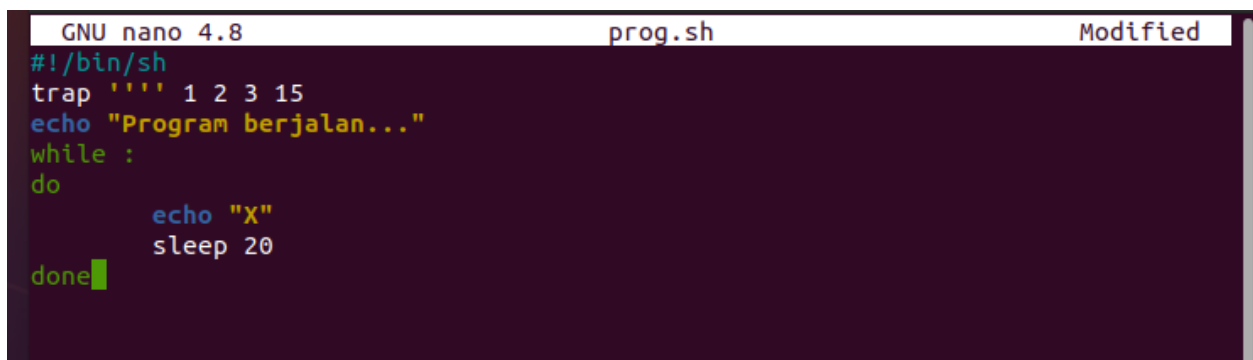
```
while :
```

```
do
```

```
echo "X"
```

```
sleep 20
```

```
done
```



```
GNU nano 4.8          prog.sh          Modified
#!/bin/sh
trap '' 1 2 3 15
echo "Program berjalan..."
while :
do
    echo "X"
    sleep 20
done
```

⇒ Pada perintah diatas mengedit program pada prog.sh dengan menambah perintah trap agar dapat menangkap sinyal yang dikirim



6. Jalankan program tersebut sebagai background. Coba

lakukan kill dengan nomor PID proses tersebut.

\$ ./prog.sh &

\$ kill [Nomor PID] atau

\$ kill -1 [Nomor PID] atau

\$ kill -2 [Nomor PID] atau

\$ kill -15 [Nomor PID]

```
laurinmadelau@laurinmadelau-VirtualBox:~$ kill 3458X
laurinmadelau@laurinmadelau-VirtualBox:~$ kill -1 3458
laurinmadelau@laurinmadelau-VirtualBox:~$ kill -2 3458
laurinmadelau@laurinmadelau-VirtualBox:~$ kill -15 3458
```

⇒ Pada perintah diatas menjalankan program pada prog.sh ke proses background lalu menghentikannya dengan perintah kill. kemudian program tersebut tidak dapat dihentikan.

7. Perintah kill diatas tidak akan menghentikan proses karena

dihalangi dengan perintah trap. Cobalah menggunakan

nomor sinyal 9

\$ kill -9 [Nomor PID]

```
laurinmadelau@laurinmadelau-VirtualBox:~$ kill -9 3458
[1]+  Killed                  ./prog.sh
laurinmadelau@laurinmadelau-VirtualBox:~$
```

⇒ Perintah diatas dapat memberhentikan perintah trap yang ada pada program prog.sh

#### Percobaan 4: No Hangup

1. Adakalanya sebuah proses memerlukan waktu yang cukup lama, misalnya proses sortir, sehingga perlu dilakukan sebagai proses background. Namun bila proses masih berlangsung dan kita melakukan logout, maka otomatis proses akan ikut berhenti, yang artinya proses sortir harus diulang kembali. Simulasi dari proses sortir

```
$ vi myjob.sh
```

```
#!/bin/sh
```

```
i=1
```

```
while :
```

```
do
```

```
find / -print > berkas
```

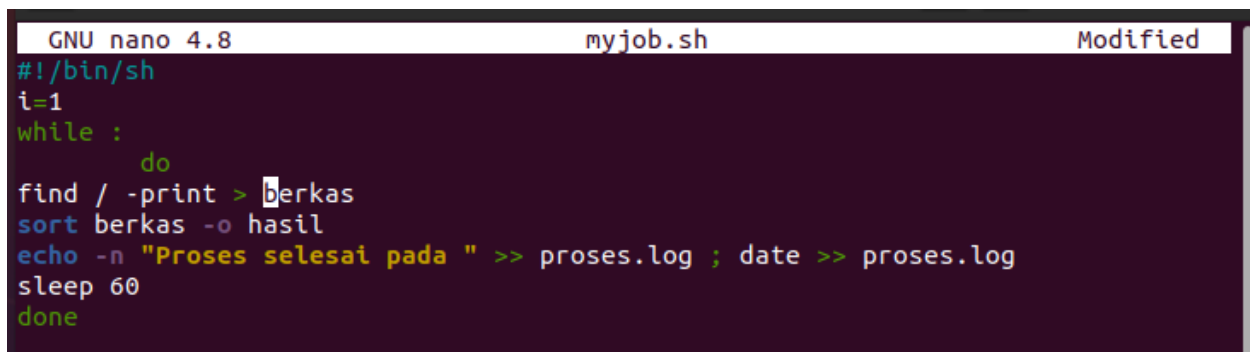
```
sort berkas -o hasil
```

```
echo -n "Proses selesai pada " >> proses.log ; date >>
```

```
proses.log
```

```
sleep 60
```

```
done
```



```
GNU nano 4.8 myjob.sh Modified
#!/bin/sh
i=1
while :
do
find / -print > berkas
sort berkas -o hasil
echo -n "Proses selesai pada " >> proses.log ; date >> proses.log
sleep 60
done
```

⇒ Membuat sebuah program pada file myjob.sh

2. Jalankan proses tersebut sebagai proses background

```
$ chmod +x myjob.sh
```

```
$ ./myjob.sh &
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ chmod +x myjob.sh
laurinmadelau@laurinmadelau-VirtualBox:~$ ./myjob.sh &
[1] 3671
laurinmadelau@laurinmadelau-VirtualBox:~$ find: '/proc/tty/driver': Permission
denied
find: '/proc/1/task/1/fd': Permission denied
find: '/proc/1/task/1/fdinfo': Permission denied
find: '/proc/1/task/1/ns': Permission denied
find: '/proc/1/fd': Permission denied
find: '/proc/1/map files': Permission denied
```

⇒ Pada gambar diatas terdapat perintah memberi izin execute pada file myjob.sh dan menjalankan ke proses background.

⇒

3. Kemudian logout dan login kembali. Periksa sampai dimana

job bekerja.

```
$ ps
```

```
ps
  PID TTY          TIME CMD
 3068 pts/0    00:00:00 bash
 3671 pts/0    00:00:00 myjob.sh
 3723 pts/0    00:00:00 sleep
 3746 pts/0    00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps
  PID TTY          TIME CMD
 3845 pts/0    00:00:00 bash
 3868 pts/0    00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

4. Gunakan nohup (NoHangup) agar job tetap berjalan meskipun pemakai logout. Catatan: fungsi ini tidak berjalan di non System

V (Linux)

```
$ ./myjob.sh &
```

```
$ nohup myjob.sh
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ./myjob.sh &
[1] 4244
laurinmadelau@laurinmadelau-VirtualBox:~$ nohup myjob.sh
nohup: ignoring input and appending output to 'nohup.out'
nohup: failed to run command 'myjob.sh': No such file or directory
laurinmadelau@laurinmadelau-VirtualBox:~$ find: '/proc/tty/driver': Permission
denied
find: '/proc/1/task/1/fd': Permission denied
find: '/proc/1/task/1/fdinfo': Permission denied
```

5. Kemudian logout dan login kembali. Periksa apakah job masih bekerja.

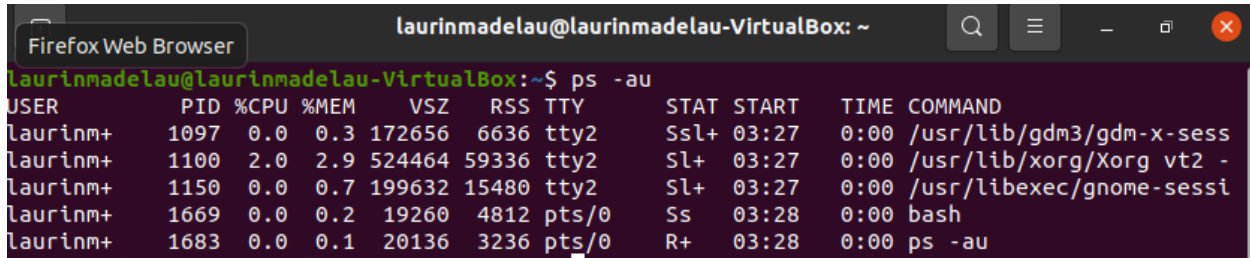
```
$ ps
```

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps
  PID TTY          TIME CMD
 4386 pts/0    00:00:00 bash
 4400 pts/0    00:00:00 ps
laurinmadelau@laurinmadelau-VirtualBox:~$
```

## E. LATIHAN

1. Login sebagai studentX dan lihat status proses, perhatikan

kolom keluaran ps -au sebagai berikut:



```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -au
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
laurinm+      1097  0.0  0.3 172656  6636 tty2      Ssl+  03:27   0:00 /usr/lib/gdm3/gdm-x-sess
laurinm+      1100  2.0  2.9 524464 59336 tty2      Sl+   03:27   0:00 /usr/lib/xorg/Xorg vt2 -
laurinm+      1150  0.0  0.7 199632 15480 tty2      Sl+   03:27   0:00 /usr/libexec/gnome-sessi
laurinm+      1669  0.0  0.2  19260  4812 pts/0     Ss    03:28   0:00 bash
laurinm+      1683  0.0  0.1  20136  3236 pts/0     R+    03:28   0:00 ps -au
```

a. Sebutkan nama-nama proses yang bukan root

- bash

- ps -au

b. Tulis PID dan COMMAND dari proses yang paling banyak menggunakan CPU time

- Tidak ada

c. Sebutkan buyut proses dan PID dari proses tersebut

- PID = 1097, STAT Ssl+

⇒ Buyut merupakan proses yang memiliki PID terkecil yang menandakan program tersebut merupakan program yang pertama dijalankan.

d. Sebutkan beberapa proses daemon

- Tidak ada

e. Pada prompt login lakukan hal-hal sebagai berikut:

\$ csh

\$ who

\$ bash

\$ ls

\$ sh

\$ ps

```
laurinmadelau@laurinmadelau-VirtualBox:~$ sh
$ ps
  PID TTY          TIME CMD
 1669 pts/0    00:00:00 bash
 1849 pts/0    00:00:00 bash
 1856 pts/0    00:00:00 sh
 1859 pts/0    00:00:00 bash
 1866 pts/0    00:00:00 sh
 1868 pts/0    00:00:00 sh
 1869 pts/0    00:00:00 bash
 1876 pts/0    00:00:00 sh
 1877 pts/0    00:00:00 ps
$
```

Sebutkan PID yang paling besar dan kemudian buat urutan-

urutan proses sampai ke PPID = 1. Lakukan ^d atau exit atau logout sampai kembali muncul login: prompt

```
laurinmadelau@laurinmadelau-VirtualBox:~$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5 101976 11608 ?        Ss   03:27   0:01 /sbin/init splash
root         2  0.0  0.0      0      0 ?        S    03:27   0:00 [kthreadd]
root         3  0.0  0.0      0      0 ?        I<   03:27   0:00 [rcu_gp]
root         4  0.0  0.0      0      0 ?        I<   03:27   0:00 [rcu_par_gp]
root         6  0.0  0.0      0      0 ?        I<   03:27   0:00 [kworker/0:0H-kblockd]
root         9  0.0  0.0      0      0 ?        I<   03:27   0:00 [mm_percpu_wq]

laurinm+   1866  0.0  0.0   2616   600 pts/0    S    03:59   0:00 sh
laurinm+   1868  0.0  0.0   2616   600 pts/0    S    04:00   0:00 sh
laurinm+   1869  0.0  0.2  19260  4884 pts/0    S    04:00   0:00 bash
laurinm+   1876  0.0  0.0   2616   600 pts/0    S    04:00   0:00 sh
laurinm+   1879  0.5  0.2  19260  4900 pts/0    S    04:02   0:00 bash
root       1887  0.0  0.4 163584  8720 ?        Ssl  04:02   0:00 /usr/lib/NetworkManager/
laurinm+   1891  0.0  0.1  20332  3480 pts/0    R+   04:02   0:00 ps -aux
```

⇒ PID paling besar adalah 1891.

2. Modifikasi program prog.sh sebagai berikut:

```
$ vi prog.sh
```

```
#!/bin/sh
```

```
trap "echo Hello Goodbye ; exit 0 " 1 2 3 15
```

```
echo "Program berjalan ..."
```

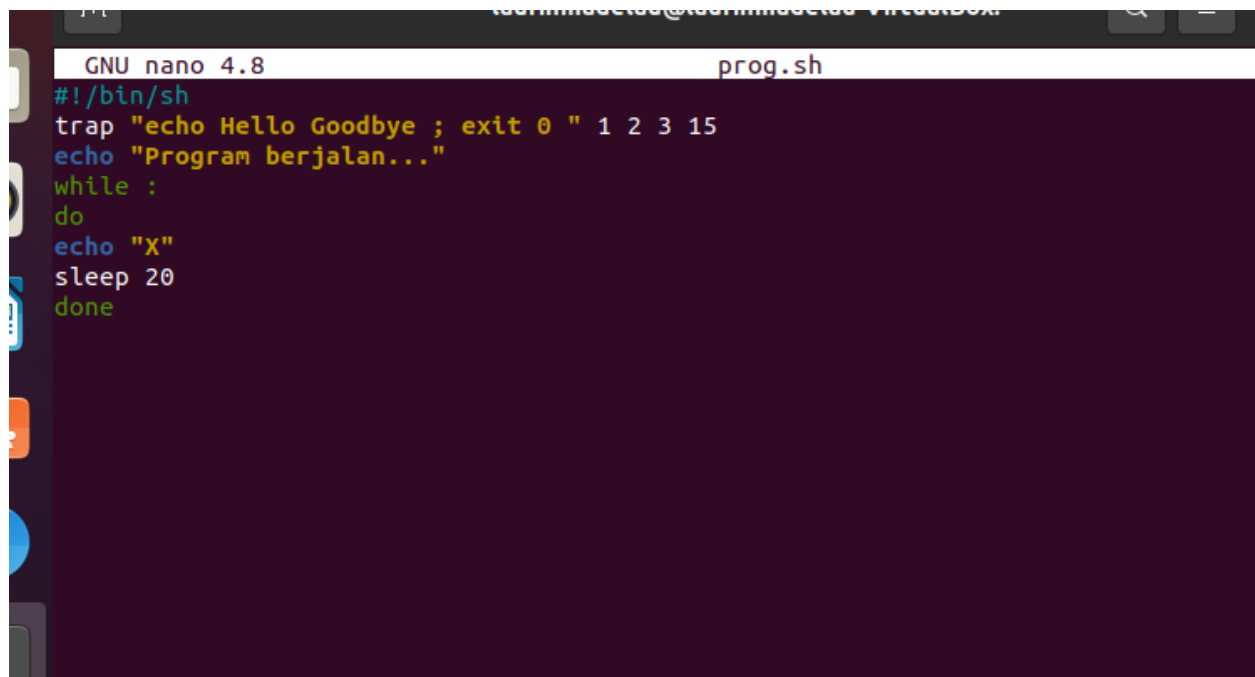
```
while :
```

```
do
```

```
echo "X"
```

```
sleep 20
```

```
done
```

A screenshot of a terminal window with a dark background. The window title is "prog.sh". The terminal shows the contents of the script being edited in nano 4.8. The script includes a shebang, a trap command, an echo statement, and a while loop with a do-while structure containing an echo and a sleep command.

```
GNU nano 4.8 prog.sh
#!/bin/sh
trap "echo Hello Goodbye ; exit 0 " 1 2 3 15
echo "Program berjalan..."
while :
do
echo "X"
sleep 20
done
```

Jalankan program tersebut sebagai background. Coba lakukan kill dengan nomor sinyal 1, 2, 3 dan 15 pada nomor PID proses tersebut. Apakah proses berhenti atau tetap berjalan? Nomor sinyal berapa yang digunakan untuk menghentikan proses di atas?

```
laurinmadelau@laurinmadelau-VirtualBox:~$ chmod +x prog.sh
laurinmadelau@laurinmadelau-VirtualBox:~$ ./prog.sh &
[1] 1932
laurinmadelau@laurinmadelau-VirtualBox:~$ Program berjalan...
X
X
X
X
```

⇒ Menjalankan program pada prog.sh ke proses background

```
X
killX
-1 1932
laurinmadelau@laurinmadelau-VirtualBox:~$ Hello Goodbye
kill -2 1932
bash: kill: (1932) - No such process
[1]+  Done                  ./prog.sh
laurinmadelau@laurinmadelau-VirtualBox:~$
```

⇒ Memberhentikan program prog.sh dengan nomor sinyal digunakan adalah -2 untuk memberhentikan program prog.sh tersebut.



3. Modifikasi program myjob.sh. Buatlah trap sedemikian rupa, sehingga bila proses tersebut dihentikan (kill), otomatis file berkas akan terhapus.

```
$ vi myjob.sh
```

```
#!/bin/sh
```

```
trap _____
```

```
i=1
```

```
while :
```

```
do
```

```
find / -print > berkas
```

```
sort berkas -o hasil
```

```
echo "Proses selesai pada 'date'" >> proses.log
```

```
sleep 60
```

```
done
```

```
$ kill -15 [Nomor PID]
```

```
$ ls -l
```

Maka file berkas tidak ada lagi!

>>>>SKIPPP<<<<