

# **JOBSHEET 6**

## **Proses dan Manajemen Proses**

**Nama:** Rizqi Bagus Andrean

**Kelas:** TI-1D

**Absen:** 25

### **Pokok Bahasan:**

- Proses pada Sistem Operasi Linux
- Manajemen Proses pada Sistem Operasi Linux

### **Tujuan Belajar:**

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- Memahami konsep proses pada sistem operasi Linux.
- Menampilkan beberapa cara menampilkan hubungan proses parent dan child.
- Menampilkan status proses dengan beberapa format berbeda.
- Melakukan pengontrolan proses pada shell.
- Memahami penjadwalan prioritas.

### **Dasar Teori:**

#### **1. Konsep Proses Pada Sistem Operasi Linux**

Proses adalah program yang sedang dieksekusi. Setiap kali menggunakan utilitas sistem atau program aplikasi dari shell, satu atau lebih proses "child" akan dibuat oleh shell sesuai perintah yang diberikan. Setiap kali instruksi diberikan pada Linux shell, maka kernel akan menciptakan sebuah proses-id. Proses ini disebut juga dengan terminology Unix sebagai sebuah Job. Proses Id (PID) dimulai dari 0, yaitu proses INIT, kemudian diikuti oleh proses berikutnya (terdaftar pada /etc/inittab). Beberapa tipe proses:

- Foreground adalah proses yang diciptakan oleh pemakai langsung pada terminal (interaktif, dialog).
- Batch adalah proses yang dikumpulkan dan dijalankan secara sekuensial (satu persatu). Proses Batch tidak diasosiasikan (berinteraksi) dengan terminal.
- Daemon adalah proses yang menunggu permintaan (request) dari proses lainnya dan menjalankan tugas sesuai dengan permintaan tersebut. Bila tidak ada request, maka program ini akan berada dalam kondisi "idle" dan tidak menggunakan waktu hitung CPU. Umumnya nama proses daemon di UNIX berakhiran d, misalnya inetd, named, popd dll.

#### **2. Sinyal**

Proses dapat mengirim dan menerima sinyal dari dan ke proses lainnya. Proses mengirim sinyal melalui instruksi “kill” dengan format

```
kill [-nomor sinyal] PID
```

### 3. Mengirim Sinyal

Mengirim sinyal adalah satu alat komunikasi antar proses, yaitu memberitahukan proses yang sedang berjalan bahwa ada sesuatu yang harus dikendalikan. Berdasarkan sinyal yang dikirim ini maka proses dapat bereaksi dan administrator/programmer dapat menentukan reaksi tersebut. Mengirim sinyal menggunakan instruksi

```
kill [-nomor sinyal] PID
```

### 4. Mengontrol Proses Pada Shell

Shell menyediakan fasilitas job control yang memungkinkan mengontrol beberapa job atau proses yang sedang berjalan pada waktu yang sama. Misalnya bila melakukan pengeditan file teks dan ingin melakukan interrupt pengeditan untuk mengerjakan hal lainnya. Bila selesai, dapat kembali (switch) ke editor dan melakukan pengeditan file teks kembali.

Job bekerja pada **foreground** atau **background**. Pada foreground hanya diperuntukkan untuk satu job pada satu waktu. Job pada foreground akan mengontrol shell - menerima input dari keyboard dan mengirim output ke layar.

Job pada background tidak menerima input dari terminal, biasanya berjalan tanpa memerlukan interaksi. Job pada foreground kemungkinan dihentikan sementara (suspend), dengan menekan [Ctrl-Z]. Job yang dihentikan sementara dapat dijalankan kembali pada foreground atau background sesuai keperluan dengan menekan “fg” atau “bg”. Sebagai catatan, menghentikan job sementara sangat berbeda dengan melakukan interrupt job (biasanya menggunakan [Ctrl-C]), dimana job yang diinterrupt akan dimatikan secara permanen dan tidak dapat dijalankan lagi.

### 5. Mengontrol Proses Lain

Perintah ps dapat digunakan untuk menunjukkan semua proses yang sedang berjalan pada mesin (bukan hanya proses pada shell saat ini) dengan format:

```
ps -fae atau  
ps -aux
```

Beberapa versi UNIX mempunyai utilitas sistem yang disebut top yang menyediakan cara interaktif untuk memonitor aktifitas sistem. Statistik secara detail dengan proses yang berjalan ditampilkan dan secara terus-menerus di-refresh. Proses ditampilkan secara terurut dari utilitas CPU. Kunci yang berguna pada top adalah

s- set update frequency  
u- display proses dari satu user  
k- kill proses (dengan PID)  
q- quit

Utilitas untuk melakukan pengontrolan proses dapat ditemukan pada sistem UNIX adalah perintah `killall`. Perintah ini akan menghentikan proses sesuai PID atau job number proses.

## **TUGAS PENDAHULUAN:**

Jawablah pertanyaan-pertanyaan di bawah ini:

1. Apa yang dimaksud dengan proses?
2. Apa yang dimaksud perintah untuk menampilkan status proses: `ps`, `pstree` ?
3. Sebutkan opsi yang dapat diberikan pada perintah `ps` !
4. Apa yang dimaksud dengan sinyal ? Apa perintah untuk mengirim sinyal ?
5. Apa yang dimaksud dengan proses foreground dan background pada job control ?
6. Apa yang dimaksud perintah-perintah penjadwalan prioritas: `top`, `nice`, `renice` ?

## **PERCOBAAN:**

1. Login sebagai user.
2. Download program C++ untuk menampilkan bilangan prima yang bernama `primes`.
3. Lakukan percobaan-percobaan di bawah ini kemudian analisa hasil percobaan.
4. Selesaikan soal-soal Latihan

## **Praktikum**

### **Percobaan 1 : Status Proses**

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.

2. Instruksi `ps` (*process status*)

\$ `ps`

```
root@bagusok:~# ps
  PID TTY          TIME CMD
 1715 pts/0        00:00:00 bash
 2445 pts/0        00:00:00 ps
root@bagusok:~# |
```

Analisa:

Perintah ini 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.

3. Untuk melihat faktor/elemen lainnya, gunakan option -u (user).

```
$ ps -u
```

```
2443 pts/0 00:00:00 ps
root@bagusok:~# ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      683  0.0  0.2   5836  4068 tty1    Ss   04:00   0:00 /bin/logi
root     1561  0.0  0.2   8264  5204 tty1    S+   04:00   0:00 -bash
root     1715  0.0  0.2   8304  5172 pts/0    Ss   04:01   0:00 -bash
root     2448  0.0  0.1   8888  3448 pts/0    R+   04:30   0:00 ps -u
root@bagusok:~# |
```

Analisa:

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 indentitas dari proses yang ditunjukan.
- %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.

4. Mencari proses yang spesifik pemakai.

```
$ ps -u <user>
```

```

root      2448  0.0  0.1  8888  3448 pts/0
root@bagusok:~# ps -u bagus
      PID TTY          TIME CMD
root@bagusok:~# |

```

Analisa:

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.

5. Mencari proses lainnya gunakan opsi a (*all*) dan au (*all user*)

```
$ ps -a
```

```

      PID TTY          TIME CMD
root@bagusok:~# ps -a
      PID TTY          TIME CMD
    1561 tty1        00:00:00 bash
    2451 pts/0        00:00:00 ps
root@bagusok:~# ps -au

```

```
$ ps -au
```

```

    2451 pts/0        00:00:00 ps
root@bagusok:~# ps -au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         683   0.0   0.2   5836   4068 tty1     Ss   04:00    0:00 /bin/login
root       1561   0.0   0.2   8264   5204 tty1     S+   04:00    0:00 -bash
root       1715   0.0   0.2   8304   5172 pts/0    Ss   04:01    0:00 -bash
root       2452   0.0   0.1   8888   3452 pts/0    R+   04:30    0:00 ps -au
root@bagusok:~#

```

Analisa:

Perintah `$ ps -a` digunakan untuk mengeksekusi perintah pada satu user saja. Sedangkan perintah `$ ps -au` digunakan untuk melihat informasi dari proses yang dijalankan oleh semua user.

6. **Logout** dan tekan **Alt+F7** untuk kembali ke mode grafis

## Percobaan 2 : Menampilkan Hubungan Proses Parent dan Child

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.

2. Ketik **ps -eH** dan tekan **Enter**. Opsi **e** memilih semua proses dan opsi **H** menghasilkan tampilan proses secara hierarki. Proses child muncul dibawah proses parent. Proses child ditandai dengan awalan beberapa spasi.

```
$ ps -eH
```

```
root      1713   0.0   0.2   8304   3172 pts/0    SS      04:01
root      2452   0.0   0.1   8888   3452 pts/0    R+      04:30
root@bagusok:~# ps -eH
  PID TTY          TIME CMD
    2 ?           00:00:00 kthreadd
    3 ?           00:00:00   rcu_gp
    4 ?           00:00:00 rcu_par_gp
    6 ?           00:00:00 kworker/0:0H
    8 ?           00:00:00 mm_percpu_wq
    9 ?           00:00:00 ksoftirqd/0
   10 ?           00:00:01 rcu_sched
   11 ?           00:00:00 migration/0
   12 ?           00:00:00 idle_inject/0
   14 ?           00:00:00 cpuhp/0
   15 ?           00:00:00 kdevtmpfs
   16 ?           00:00:00 netns
   17 ?           00:00:00 rcu_tasks_kthre
   18 ?           00:00:00 kauditd
   19 ?           00:00:00 khungtaskd
   20 ?           00:00:00 oom_reaper
   21 ?           00:00:00 writeback
   22 ?           00:00:00 kcompactd0
```

Analisa:

Perintah diatas sama fungsinya dengan perintah `$ ps` pada perintah-perintah yang telah dijalankan sebelumnya yang perbedaanya hanya pada opsi yang ditambahkan setelahnya dimana pada perintah `$ ps` digabungkan dengan opsi `-eH`. Dan ketika perintah `$ ps -eH` ini dieksekusi dengan cara menekan Enter, maka prosesnya akan berjalan dengan cara membaca terlebih dahulu perintah `$ ps` yang kemudian dilanjutkan dengan membaca opsi `e` yang berfungsi memilih semua proses dan opsi `H` yang berfungsi menghasilkan tampilan proses secara hierarki.

3. Ketik **ps -e f** dan tekan **Enter**. Tampilan serupa dengan langkah 2. Opsi **-f** akan menampilkan status proses dengan karakter grafis (`\` dan `_`)

```
$ ps -e f
```

```

root@bagusok:~#
root@bagusok:~# ps -e f
  PID TTY          STAT TIME  COMMAND
    2 ?            S      0:00 [kthreadd]
    3 ?            I<      0:00  \_ [rcu_gp]
    4 ?            I<      0:00  \_ [rcu_par_gp]
    6 ?            I<      0:00  \_ [kworker/0:0H]
    8 ?            I<      0:00  \_ [mm_percpu_wq]
    9 ?            S      0:00  \_ [ksoftirqd/0]
   10 ?            I      0:01  \_ [rcu_sched]
   11 ?            S      0:00  \_ [migration/0]
   12 ?            S      0:00  \_ [idle_inject/0]
   14 ?            S      0:00  \_ [cpuhp/0]
   15 ?            S      0:00  \_ [kdevtmpfs]
   16 ?            I<      0:00  \_ [netns]

```

Analisa:

Perintah ini serupa dengan tampilan pada percobaan ketiga diatas, yang berbeda adalah pada opsi yang ditambahkan setelah opsi `-e`. Dimana pada perintah ini ditambahkan opsi `f` yang berfungsi untuk mengetahui STAT (keadaan) dari sebuah proses itu yang biasanya ditandai dengan simbol S (sleeping) atau R (Running).

4. Ketik **pstree** dan tekan **Enter**.

```
$ pstree
```

```

1556 ?            S      0:00  \_ (sd-pam)
root@bagusok:~# pstree
systemd--ModemManager--2*[{ModemManager}]
      |--accounts-daemon--2*[{accounts-daemon}]
      |--atd
      |--cron
      |--dbus-daemon
      |--login--bash
      |--master--pickup
                  |--qmgr
      |--multipathd--6*[{multipathd}]
      |--networkd-dispat
      |--polkitd--2*[{polkitd}]
      |--rsyslogd--3*[{rsyslogd}]
      |--snapd--7*[{snapd}]
      |--sshd--sshd--bash--pstree
      |--systemd--(sd-pam)
      |--systemd-journal
      |--systemd-logind
      |--systemd-network
      |--systemd-resolve
      |--systemd-timesyn--{systemd-timesyn}
      |--systemd-udev
      |--udisksd--4*[{udisksd}]

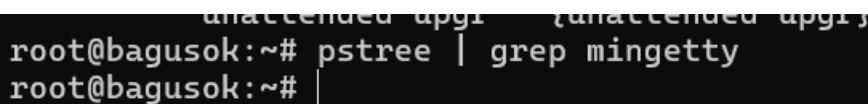
```

Analisa:

Gambar diatas tampak struktur berupa sebuah pohon atau diagram. Yang menyatakan sistem dalam bentuk hirarki parent/child. Proses parent di sebelah kiri proses child. Sebagai contoh proses `init` sebagai parent (*ancestor*) dari semua proses pada sistem. Beberapa child dari `init` mempunyai child. Proses `login` mempunyai proses `bash` sebagai child. Proses `bash` mempunyai proses child `startx`. Proses `startx` mempunyai child `xinit` dan seterusnya.

5. Ketik **pstree | grep mingetty** dan tekan **Enter**.

```
$ pstree | grep mingetty
```



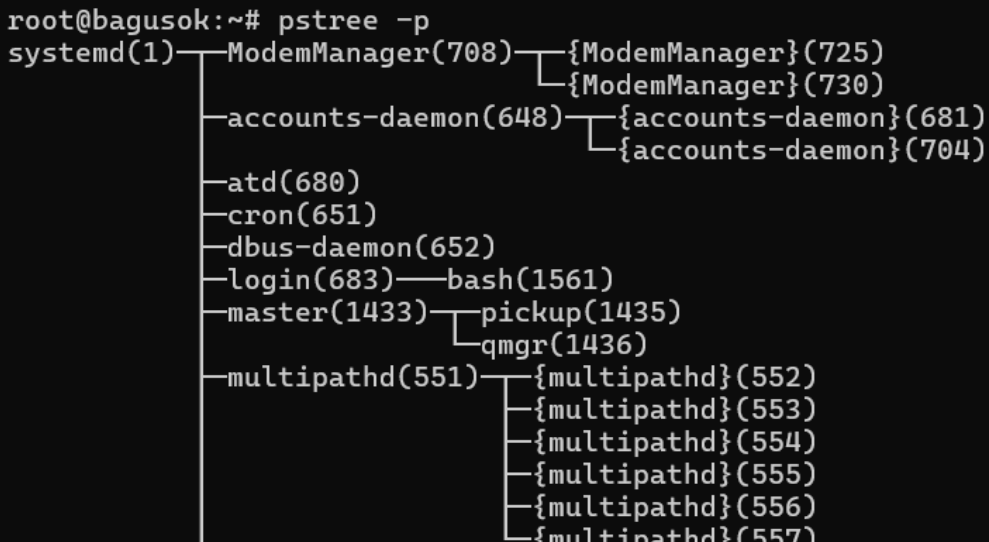
```
root@bagusok:~# pstree | grep mingetty
root@bagusok:~#
```

Anaisa:

Perintah ini digunakan untuk menampilkan semua proses *mingetty* yang berjalan pada system yang berupa *console virtual*. Selain menampilkan semua proses, proses dikelompokkan dalam satu baris dengan suatu angka sebagai jumlah proses yang berjalan.

6. Untuk melihat semua PID untuk proses gunakan opsi **-p**.

```
$ pstree -p
```



```
root@bagusok:~# pstree -p
systemd(1)─ModemManager(708)─{ModemManager}(725)
               │               │{ModemManager}(730)
               └─accounts-daemon(648)─{accounts-daemon}(681)
                                   │{accounts-daemon}(704)
               └─atd(680)
               └─cron(651)
               └─dbus-daemon(652)
               └─login(683)─bash(1561)
               └─master(1433)─pickup(1435)
                               │qmgr(1436)
               └─multipathd(551)─{multipathd}(552)
                                   │{multipathd}(553)
                                   │{multipathd}(554)
                                   │{multipathd}(555)
                                   │{multipathd}(556)
                                   │{multipathd}(557)
```

Analisa:

Proses tampilan dari perintah ini serupa dengan tampilan pada perintah yang dilakukan pada proses `$ pstree` dimana data ditampilkan menyerupai sebuah struktur diagram atau pohon. Yang pada proses ini hanya ditambahkan dengan informasi mengenai PID dari proses yang digunakan dengan menambahkan Opsi `-p`.



7. Untuk menampilkan proses dan ancestor yang tercetak tebal gunakan opsi **-h**.

```
$ pstree -h
```

```
└─unattended-upgr(722)──{unattended-upgr}(758)
root@bagusok:~# pstree -h
systemd─ModemManager──2*[{ModemManager}]
        └─accounts-daemon──2*[{accounts-daemon}]
        atd
        cron
        dbus-daemon
        login──bash
        master└─pickup
                └─qmgr
        multipathd──6*[{multipathd}]
        networkd-dispat
        polkitd──2*[{polkitd}]
        rsyslogd──3*[{rsyslogd}]
        snapd──7*[{snapd}]
        sshd──sshd──bash──pstree
        systemd──(sd-pam)
        systemd-journal
        systemd-logind
        systemd-network
        systemd-resolve
        systemd-timesyn──{systemd-timesyn}
        systemd-udev
        udiskd──11*[{udiskd}]
```

Analisa:

Perintah `$ pstree` yang kemudian ditambahkan opsi `-h` berfungsi untuk menampilkan proses dan ancestor dengan cara ditampilkan atau dicetak tebal.

### Percobaan 3 : Menampilkan Status Proses dengan Berbagai Format

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.

2. Ketik **ps -e | more** dan tekan **Enter**.

```
$ ps e | more
```

```
For more details see ps(1).
root@bagusok:~# ps e
  PID TTY          STAT       TIME COMMAND
   683 tty1        Ss          0:00 /bin/login -p -- PATH=/usr/local/sbin:/usr/loca
  1561 tty1        S+          0:00 -bash INVOCATION_ID=70183487603e4875b23ea384297
  1715 pts/0      Ss          0:00 -bash USER=root LOGNAME=root HOME=/root PATH=/u
  2466 pts/0      R+          0:00 ps e SHELL=/bin/bash PWD=/root LOGNAME=root XDG
root@bagusok:~# |
```

Jika halaman penuh terlihat prompt --More--di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

Analisa:

Opsi **-e** menampilkan semua proses dalam bentuk 4 kolom : PID, TTY, TIME dan CMD.

3. Ketik **ps ax | more** dan tekan **Enter**.

```
$ ps ax | more
```

```
1713 pts/0    Ss   0:00 -bash USER=root LOGNAME=root HOME=/
2466 pts/0    R+   0:00 ps e SHELL=/bin/bash PWD=/root LOGN
root@bagusok:~# ps ax | more
  PID TTY          STAT TIME COMMAND
    1 ?           Ss    0:09 /sbin/init maybe-ubiquity
    2 ?           S      0:00 [kthreadd]
    3 ?           I<    0:00 [rcu_gp]
    4 ?           I<    0:00 [rcu_par_gp]
    6 ?           I<    0:00 [kworker/0:0H]
    8 ?           I<    0:00 [mm_percpu_wq]
    9 ?           S      0:00 [ksoftirqd/0]
   10 ?          I      0:01 [rcu_sched]
   11 ?           S      0:00 [migration/0]
   12 ?           S      0:00 [idle_inject/0]
   14 ?           S      0:00 [cpuhp/0]
   15 ?           S      0:00 [kdevtmpfs]
   16 ?          I<    0:00 [netns]
   17 ?           S      0:00 [rcu_tasks_kthre]
   18 ?           S      0:00 [kauditd]
   19 ?           S      0:00 [khungtaskd]
   20 ?           S      0:00 [oom_reaper]
   21 ?          I<    0:00 [writeback]
   22 ?           S      0:00 [kcompactd0]
```

Jika halaman penuh terlihat prompt --More--di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

Analisa:

Opsi **a** berfungsi untuk menampilkan semua proses yang dihasilkan terminal (TTY), yang dilanjutkan dengan membaca Opsi **x** yang akan menampilkan semua proses yang tidak dihasilkan terminal. Secara logika opsi **ax** ini sama dengan opsi **-e**, dimana terdapat 5 kolom : PID, TTY, STAT, TIME dan COMMAND.

```
$ ps ef | more
```

```
root@bagusok:~# ps ef | more
  PID TTY          STAT TIME COMMAND
  1715 pts/0      Ss   0:00 -bash USER=root LOGNAME=root HOME=/root PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin SHELL=/bin/bash TERM=xterm-256color XDG_SESSION_ID=3 XDG_RUNTIME_DIR=/run/user/0 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/0/bus XDG_SESSION_TYPE=tty XDG_SESSION_CLASS=user MOTD_SHOWN=pam LANG=en_US.UTF-8 SSH_CLIENT=192.168.64.90 63184 22 SSH_CONNECTION=192.168.64.90 63184 192.168.64.94 22 SSH_TTY=/dev/pts/0
  2469 pts/0      R+   0:00 \_ ps ef SHELL=/bin/bash PWD=/root LOGNAME=root XDG_SESSION_TYPE=tty MOTD_SHOWN=pam HOME=/root LANG=en_US.UTF-8 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpq=01;35:*.mpeg=01;35:*.m2v=01;3
```

Jika halaman penuh terlihat prompt --More--di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

Analisa:

Ketika perintah `$ ps ef | more` dieksekusi, maka prosesnya akan diawali dengan membaca `$ ps` dilanjutkan dengan membaca `ef | more`. Opsi `-ef` akan menampilkan semua proses dalam format daftar penuh.

5. Ketik **ps-eo pid, cmd | more** dan tekan **Enter**.

```
$ ps eo pid,cmd | more
```

```

root@bagusok:~# ps eo pid,cmd | more
  PID CMD
  683 /bin/login -p -- PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/snap/bin INVOCATION_ID=70183487603e4875b23ea38429766e10 TERM
=linux JOURNAL_STREAM=9:24749
 1561 -bash INVOCATION_ID=70183487603e4875b23ea38429766e10 TERM=linux JOUR
NAL_STREAM=9:24749 HOME=/root SHELL=/bin/bash USER=root LOGNAME=root PATH=/u
sr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/l
ocal/games:/snap/bin MOTD_SHOWN=pam LANG=en_US.UTF-8 MAIL=/var/mail/root XDG
_SESSION_ID=1 XDG_RUNTIME_DIR=/run/user/0 DBUS_SESSION_BUS_ADDRESS=unix:path
=/run/user/0/bus XDG_SESSION_TYPE=ttty XDG_SESSION_CLASS=user XDG_SEAT=seat0
XDG_VTNR=1 HUSHLOGIN=FALSE
 1715 -bash USER=root LOGNAME=root HOME=/root PATH=/usr/local/sbin:/usr/lo
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
SHELL=/bin/bash TERM=xterm-256color XDG_SESSION_ID=3 XDG_RUNTIME_DIR=/run/us
er/0 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/0/bus XDG_SESSION_TYPE=ttty
XDG_SESSION_CLASS=user MOTD_SHOWN=pam LANG=en_US.UTF-8 SSH_CLIENT=192.168.6
4.90 63184 22 SSH_CONNECTION=192.168.64.90 63184 192.168.64.94 22 SSH_TTY=/d
ev/pts/0
 2485 ps eo pid,cmd SHELL=/bin/bash PWD=/root LOGNAME=root XDG_SESSION_TYP
E=ttty MOTD_SHOWN=pam HOME=/root LANG=en_US.UTF-8 LS_COLORS=rs=0:di=01:34:ln=

```

Jika halaman penuh terlihat prompt --More--di bagian bawah screen, tekan **q** untuk kembali ke prompt perintah.

Analisa:

Ketika perintah `$ ps -eo pid,cmd | more` dieksekusi, maka prosesnya akan diawali dengan membaca `$ ps` dilanjutkan dengan membaca `-eo pid,cmd | more`. Opsi `-eo` akan menampilkan semua proses dalam format sesuai definisi user yaitu terdiri dari kolom PID dan CMD.

6. Ketik **`ps -eo pid,ppid,%mem,cmd | more`** dan tekan **Enter**.

```
$ ps -eo pid,ppid,%mem,cmd | more
```

```

eo: command not found
root@bagusok:~# ps eo pid,cmd | more
PID CMD
683 /bin/login -p -- PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/snap/bin INVOCATION_ID=70183487603e4875b23ea38429766e10 TERM
=linux JOURNAL_STREAM=9:24749
1561 -bash INVOCATION_ID=70183487603e4875b23ea38429766e10 TERM=linux JOUR
NAL_STREAM=9:24749 HOME=/root SHELL=/bin/bash USER=root LOGNAME=root PATH=/u
sr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/l
ocal/games:/snap/bin MOTD_SHOWN=pam LANG=en_US.UTF-8 MAIL=/var/mail/root XDG
_SESSION_ID=1 XDG_RUNTIME_DIR=/run/user/0 DBUS_SESSION_BUS_ADDRESS=unix:path
=/run/user/0/bus XDG_SESSION_TYPE=tty XDG_SESSION_CLASS=user XDG_SEAT=seat0
XDG_VTNR=1 HUSHLOGIN=FALSE
1715 -bash USER=root LOGNAME=root HOME=/root PATH=/usr/local/sbin:/usr/lo
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
SHELL=/bin/bash TERM=xterm-256color XDG_SESSION_ID=3 XDG_RUNTIME_DIR=/run/us
er/0 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/0/bus XDG_SESSION_TYPE=tty
XDG_SESSION_CLASS=user MOTD_SHOWN=pam LANG=en_US.UTF-8 SSH_CLIENT=192.168.6
4.90 63184 22 SSH_CONNECTION=192.168.64.90 63184 192.168.64.94 22 SSH_TTY=/d
ev/pts/0
2499 ps eo pid,cmd SHELL=/bin/bash PWD=/root LOGNAME=root XDG_SESSION_TYP
E=tty MOTD_SHOWN=pam HOME=/root LANG=en_US.UTF-8 LS_COLORS=rs=0:di=01;34:ln=
01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:m
i=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01

```

Analisa:

Ketika perintah `$ ps -eo pid,ppid,%mem,cmd | more` dieksekusi, maka prosesnya akan diawali dengan membaca `$ ps` dilanjutkan dengan membaca `-eo pid,ppid,%mem,cmd | more`. Perintah ini akan menampilkan kolom PID, PPID dan %MEM. PPID adalah proses ID dari proses parent. %MEM menampilkan persentasi memory system yang digunakan proses. Jika proses hanya menggunakan sedikit memory system akan ditampilkan 0.

7. Logout dan tekan Alt+F7 untuk kembali ke mode grafis

## Percobaan 4 : Mengontrol proses pada shell

1. Pindah ke *command line terminal* (tty2) dengan menekan **Ctrl+Alt+F2** dan login ke terminal sebagai user.

2. Gunakan perintah `yes` yang mengirim output yang tidak pernah berhenti

```
$ yes
```

Untuk menghentikannya gunakan **Ctrl-C**.

Analisa:

Perintah `yes` ini digunakan untuk mengirim output yang tidak pernah berhenti.

3. Belokkan standart output ke `/dev/null`

```
$ yes > /dev/null
```

```
root@bagusok:~# yes > /dev/null
```

Untuk menghentikannya gunakan **Ctrl-C**.

Analisa:

Perintah `$ yes > /dev/null` ini digunakan untuk membelokan standart output dari `yes` ke `/dev/null`.

4. Salah satu cara agar perintah `yes` tetap dijalankan tetapi shell tetap digunakan untuk hal yang lain dengan meletakkan proses pada *background* dengan menambahkan karakter `&` pada akhir perintah.

```
$ yes > /dev/null &
```

```
^C
root@bagusok:~# yes > /dev/null &
[1] 2503
root@bagusok:~#
```

Angka dalam "[ ]" merupakan **job number** diikuti PID.

Analisa:

Perintah `yes` tetap dijalankan tetapi shell tetap digunakan untuk hal yang lain dengan meletakkan proses pada *background* dengan menambahkan karakter `&` pada akhir perintah.

5. Untuk melihat status proses gunakan perintah `jobs`.

```
$ jobs
```

```
root@bagusok:~# jobs
[1]+  Terminated                  yes > /dev/null
root@bagusok:~#
```

Analisa:

Perintah ini digunakan untuk melihat status proses yang telah digunakan.

6. Untuk menghentikan job, gunakan perintah `kill` diikuti *job number* atau PID proses. Untuk identifikasi job number, diikuti prefix dengan karakter `""`.

```
$ kill %<nomor job> contoh: kill %14.
```

```
root@bagusok:~# kill 1
```

Analisa:

Perintah ini digunakan untuk menghentikan job untuk identifikasi job number, diikuti prefix dengan karakter "%".

#### 7. Lihat status job setelah diterminasi

```
$ jobs
```

Analisa:

Perintah ini digunakan untuk melihat status job setelah diterminasi.

### Percobaan 5 : Menghentikan dan memulai kembali job

1. Cara lain meletakkan job pada *background* dengan memulai job secara normal (pada *foreground*), stop job dan memulai lagi pada *background*

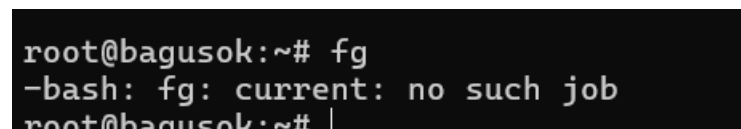
```
$ yes > /dev/null
```

Analisa:

Perintah `$ yes > /dev/null` digunakan untuk memulai job secara normal (pada *foreground*), job dapat di hentikan sementara (*suspend*), bukan menghentikannya (*terminate*), tetapi menghentikan sementara job sampai di restart. Untuk menghentikan sementara job gunakan **Ctrl-Z**.

2. Untuk restart job pada *foreground*, gunakan perintah *fg*.

```
$ fg
```



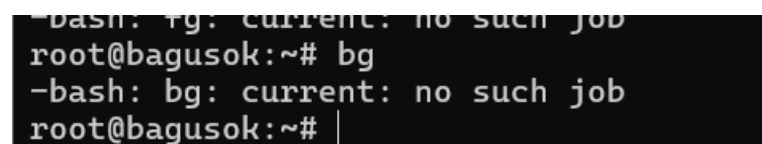
```
root@bagusok:~# fg
-bash: fg: current: no such job
root@bagusok:~# |
```

Analisa:

Perintah `$ fg` digunakan untuk restart job pada *foreground* artinya memulai kembali job yang telah di hentikan sementara (*suspend*) pada *foreground*.

3. Shell akan menampilkan nama perintah yang diletakkan di *foreground*. Stop job lagi dengan **Ctrl-Z**. Kemudian gunakan perintah *bg* untuk meletakkan job pada *background*.

```
$ bg
```



```
-bash: fg: current: no such job
root@bagusok:~# bg
-bash: bg: current: no such job
root@bagusok:~# |
```

```
$ fg
```

Analisa:

Perintah ini akan men-suspend job `yes >/dev/null`, kemudian memindahkannya ke *background* proses yang berarti proses atau job tersebut tidak berhenti akan tetapi terus berjalan di belakang layar. Job tidak bisa dihentikan dengan **Ctrl-Z** karena job berada pada *background*. Untuk menghentikannya, letakkan job pada *foreground* dengan `fg` dan kemudian hentikan sementara dengan **Ctrl-Z**.

4. Job pada *background* dapat digunakan untuk menampilkan teks pada terminal, dimana dapat diabaikan jika mencoba mengerjakan job lain.

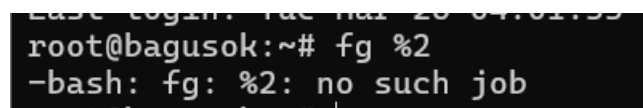
```
$ yes &
```

Analisa:

Perintah `$ yes &` ini berjalan pada job background yang telah buat tadi. Untuk menghentikannya tidak dapat menggunakan **Ctrl-C**. Job harus dipindah ke *foreground*, baru dihentikan dengan cara tekan **fg** dan tekan **Enter**, kemudian dilanjutkan dengan **Ctrl-Z** untuk menghentikan sementara.

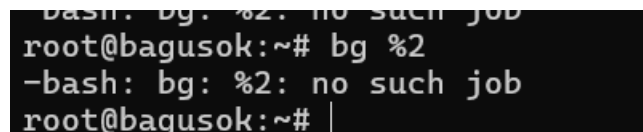
5. Apabila ingin menjalankan banyak job dalam satu waktu, letakkan job pada *foreground* atau *background* dengan memberikan job ID

```
$ fg %2 atau $ %2
```



```
root@bagusok:~# fg %2
-bash: fg: %2: no such job
```

```
$ bg %2
```



```
root@bagusok:~# bg %2
-bash: bg: %2: no such job
root@bagusok:~#
```

Analisa:

Perintah `$ fg %2` dan `$ bg %2` disini berguna untuk menjalankan banyak job dalam waktu bersamaan. Job – job yang sedang tidak dipakai dimasukkan ke proses background dan untuk mengenalinya kita menggunakan nomor job ID karena tidak ada job yang memiliki nomor ID sama.

6. Tekan **fg** dan tekan **Enter**, kemudian dilanjutkan dengan **Ctrl-Z** untuk menghentikan sementara.

Analisa:

Pada saat menekan **fg**, muncul hasil dari perintah `$yes &` tadi. Jadi fungsi perintah **fg** disini adalah untuk memanggil proses yang berjalan di background. Dan kita bisa menghentikannya dengan menekan **Ctrl + Z**.

7. Lihat job dengan perintah `ps -fae` dan tekan **Enter**. Kemudian hentikan proses dengan perintah `kill`.



```
$ ps -fae
```

```
-bash: bg: %2: no such job
root@bagusok:~# ps -fae
UID          PID    PPID  C STIME TTY          TIME CMD
root          1        0  0  03:59 ?        00:00:09 /lib/systemd/systemd
root          2        0  0  03:59 ?        00:00:00 [kthreadd]
root          3        2  0  03:59 ?        00:00:00 [rcu_gp]
root          4        2  0  03:59 ?        00:00:00 [rcu_par_gp]
root          6        2  0  03:59 ?        00:00:00 [kworker/0:0H]
root          8        2  0  03:59 ?        00:00:00 [mm_percpu_wq]
root          9        2  0  03:59 ?        00:00:00 [ksoftirqd/0]
root         10        2  0  03:59 ?        00:00:01 [rcu_sched]
root         11        2  0  03:59 ?        00:00:00 [migration/0]
```

```
$ kill -9 <NomorPID>
```

```
root@bagusok:~# kill -9 1
root@bagusok:~# |
```

Analisa:

Perintah `$ ps -fae` digunakan untuk menampilkan secara lengkap seluruh proses yang sedang berjalan beserta detailnya termasuk proses dari perintah `$ ps -fae` sendiri. Sedangkan perintah `$ kill -9 <NomorPID>` digunakan untuk menghentikan atau terminate suatu proses, berdasarkan nomor PID proses yang ingin dihentikan.

8. **Logout** dan tekan **Alt+F7** untuk kembali ke mode grafis

## Percobaan 6 : Percobaan dengan Penjadwalan Prioritas

1. Login sebagai root.
2. Buka 3 terminal, tampilkan pada screen yang sama.
3. Pada setiap terminal, ketik **PS1 = "\w:"** diikuti **Enter**. **\w** menampilkan path pada direktori home.

Analisa:

Perintah `$ PS1="\w:"` digunakan untuk masuk ke directory home dari user root.

4. Karena login sebagai root, maka akan ditampilkan `~`: pada setiap terminal. Untuk setiap terminal ketik **pwd** dan tekan **Enter** untuk melihat bahwa Anda sedang berada pada direktori `/root`.

Analisa:

Bila posisi masih berada di `/home/<user>`, maka gunakan perintah `$cd ~` untuk masuk ke root, kemudian untuk mengeceknya gunakan perintah `$pwd`.

```
root@bagusok:~# PS1="\w:"  
"w:"  
"w:"  
"w:"  
"w:"  
"w:"  
"w:"  
"w:"  
"w:"pwd  
/root  
"w:"^C  
"w:"^C  
"w:"^C  
"w:"|
```

5. Buka terminal lagi (keempat), atur posisi sehingga keempat terminal terlihat pada screen.

6. Pada terminal keempat, ketik **top** dan tekan **Enter**. Maka program **top** akan muncul. Ketik **i**. **Top** akan menampilkan proses yang aktif. Ketik **lmt**. **Top** tidak lagi menampilkan informasi pada bagian atas dari screen. Pada percobaan ini, terminal ke empat sebagai jendela **Top**.

Analisa:

Perintah **top** digunakan untuk mengetahui semua rincian proses yang berjalan, dan beberapa fungsi lainnya. Mengetikkan 'i' pada window **top** akan menampilkan proses yang sedang aktif. Mengetikkan 'lmt' untuk menghilangkan atau tidak menampilkan informasi pada bagian atas dari tampilan **top**.

```
top - 04:40:25 up 40 min,  2 users,  load average: 0.03, 0.10, 0.05
Tasks: 101 total,  2 running,  99 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.0 us,  0.7 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0
MiB Mem : 1971.6 total, 1424.0 free,  151.8 used,  395.8 buff/cach
MiB Swap: 1956.0 total, 1956.0 free,  0.0 used. 1664.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
2443	root	20	0	0	0	0	I	0.7	0.0	0:03.26
10	root	20	0	0	0	0	R	0.3	0.0	0:01.53
2583	root	20	0	13908	9060	7608	S	0.3	0.4	0:00.10
2688	root	20	0	9288	3808	3276	R	0.3	0.2	0:00.01
1	root	20	0	20828	11132	8116	S	0.0	0.6	0:09.57
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
9	root	20	0	0	0	0	S	0.0	0.0	0:00.58
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.13
12	root	-51	0	0	0	0	S	0.0	0.0	0:00.00
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00
16	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00
23	root	25	5	0	0	0	S	0.0	0.0	0:00.00
24	root	39	19	0	0	0	S	0.0	0.0	0:00.00
70	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
71	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
72	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
73	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
74	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
75	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
76	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
77	root	0	-20	0	0	0	I	0.0	0.0	0:00.00

7. Pada terminal 1, bukalah program executable C++ dengan mengetik program yes dan tekan **Enter**.

Analisa:

Perintah \$ yes diatas untuk membuat proses baru dan itu diulang-ulang terus tidak berhenti.

8. Ulangi langkah 7 untuk terminal 2.

Analisa:

Perintah ini mengulangi langkah nomor 7 pada terminal kedua. Dan hasilnya sama yaitu yes untuk membuat proses baru akan diulang terus menerus.

9. Jendela **Top** akan menampilkan dua program yes sebagai proses yang berjalan. Nilai %CPU sama pada keduanya. Hal ini berarti kedua proses mengkonsumsi waktu proses yang sama dan berjalan sama cepat. PID dari kedua proses akan berbeda, 3241 dan 3242. Kemudian gunakan terminal 3 (yang tidak menjalankan primes

maupun Jendela **Top**) dan ketik **renice 19 <PID terimnal 1>** (contoh : **renice 19 3241**) dan diikuti **Enter**. Hal ini berarti mengganti penjadwalan prioritas dari proses ke 19.

Analisa:

PID dua program yang telah dijalankan tadi adalah 3241 dan 3242. Dua proses yang sama akan menggunakan sumber daya yang hampir sama besarnya (mendekati). Perintah \$ **renice 19 3241** berarti mengganti penjadwalan prioritas dari proses ke 19, dan NI berubah yang tadinya 0 menjadi 19.

10. Tunggu beberapa saat sampai program **top** berubah dan terlihat pada jendela **Top**. Pada kolom **STAT** memperlihatkan **N** untuk proses 3241. Hal ini berarti bahwa penjadwalan prioritas untuk proses 3241 lebih besar (lebih lambat) dari 0. Proses 3242 berjalan lebih cepat.

Analisa:

Setelah prioritasnya diubah menjadi 19, proses 3241 berjalan lebih lambat dari proses 3242. Ini disebabkan proses 3241 mendapatkan penjadwalan prioritas lebih besar dari proses 3242.

11. Program **top** juga mempunyai fungsi yang sama dengan program **renice**. Pilih Jendela **Top** dan tekan **r**. Program **top** terdapat prompt **PID to renice:** tekan **3241** (ingat bahwa Anda harus mengganti 3241 dengan PID Anda sendiri) dan tekan **Enter**. Program **top** memberikan prompt **Renice PID 3241 to value:** tekan **-19** dan tekan **Enter**.

Analisa:

Program **top** juga memiliki fungsi yang sama dengan program **renice**. Jadi dapat diubah konfigurasinya proses melalui program **renice** maupun **top**.

12. Tunggu beberapa saat sampai **top** berubah dan lihat nilai % CPU pada kedua proses. Sekarang proses 3241 lebih cepat dari proses 3242. Kolom status menunjukkan < pada proses 3241 yang menunjukkan penjadwalan prioritas lebih rendah (lebih cepat) dari nilai 0.

Analisa:

Setelah prioritas proses 3241 diubah menjadi -19, proses 3241 menjadi lebih cepat daripada 3242.

13. Pilih terminal 3 (yang sedang tidak menjalankan yesatau program **top**) dan ketik **nice -n -10 yes** dan tekan **Enter**. Tunggu beberapa saat agar program **top** berubah dan akan terlihat proses **primes** ketiga. Misalnya PID nya 3241. Opsi -10 berada pada kolom NI (penjadwalan prioritas).

Analisa:

Perintah \$ **nice -n -10 yes** digunakan untuk membuat proses baru, dan opsi -10 merupakan penentuan prioritas dari proses tersebut. PID dari proses yang dibuat adalah 3241, dan disana terlihat pada kolom NI terdapat angka 19.

14. Jangan menggunakan mouse dan keyboard selama 10 detik. Program **top** menampilkan proses yang aktif selain program **yes**. Maka akan terlihat proses **top** terdaftar tetapi %CPU

kecil (dibawah 1.0) dan konsisten. Juga terlihat proses berhubungan dengan dekstop grafis seperti X, panel dll.

Analisa:

Saat mouse dan keyboard diam, penggunaan sumber daya oleh program `top` kecil. Namun ketika mouse mulai digerakkan atau ada perubahan posisi komponen semisal pointer, dll, proses `top` memakan sumber daya lebih banyak daripada saat diam tadi, salah satu alasannya adalah proses 3241 berjalan pada prioritas yang tinggi.

15. Pindahkan mouse sehingga kursor berubah pada screen dan lihat apa yang terjadi dengan tampilan `top`. Proses tambahan akan muncul dan nilai %CPU berubah sebagai bagian grafis yang bekerja. Satu alasan adalah bahwa proses 4107 berjalan pada penjadwalan prioritas tinggi. Pilih jendela Top, ketik `r`. PID to renice: muncul prompt. Ketik 3241 (ubahlah 3241 dengan PID Anda) dan tekan Enter. Renice PID 3241 to value: muncul prompt. Ketik 0 dan tekan Enter. Sekarang pindahkan mouse ke sekeliling screen. Lihat perubahannya.

Analisa:

Saat kita memindahkan kursor mouse, beberapa proses yang muncul tadi penggunaan %CPU nya berubah semua dan cenderung bertambah. Kemudian muncul lagi proses lain dengan penggunaan %CPU lumayan banyak. Kita dapat mengubah konfigurasi suatu proses melalui perintah – perintah yang terdapat pada proses `top`. Dan ternyata setelah proses 3241 diubah prioritasnya menjadi 0, penggunaan sumber daya oleh terminal `top` menjadi lebih stabil (tidak banyak perubahan) walaupun mouse digerakkan ke sekeliling screen.

16. Tutup semua terminal window.

17. Logout dan login kembali sebagai user

## Kesimpulan:

- Proses adalah program yang sedang berjalan atau sebuah kinerja yang dijalankan dalam komputer yang sedang dieksekusi, dimana setiap kali kita membuat atau menjalankan sebuah proses maka akan dibuatkan sebuah tanda terhadap proses yang kita jalankan tersebut, tanda yang dijalankan dapat berupa nomor id, nama dari proses itu sendiri, jumlah kapasitas penyimpanan yang digunakan dan waktu yang digunakan untuk mengaksesnya.
- Dalam sistem operasi linux proses disimbolkan dengan PID, TTY, TIME, CMD dan masih banyak lagi sesuai dengan perintah atau proses yang dijalankan. Setelah kita menjalankan sebuah proses, maka pasti kita juga ingin menghentikanya dalam system operasi linux untuk menghentikan atau membunuh sebuah proses kita dapat menggunakan perintah `kill` yang diikuti dengan nomor Id atau PID dari proses.
- Setiap kali instruksi atau perintah yang diberikan pada Shell Linux, kernel secara otomatis akan menciptakan proses- id. Proses ini disebut juga dalam terminologi UNIX sebagai JOB. Proses - proses sistem terbagi dalam tiga tipe utama, yaitu (a) Interactive: Diprakarsai oleh sebuah shell dan berjalan dalam foreground dan background. (b) Batch: Secara tipikal merupakan sebuah seri dari proses-proses yang dijadwalkan untuk dieksekusi pada suatu waktu tertentu. (c) Daemon : Secara tipikal diinisialisasi saat boot untuk membentuk fungsi-fungsi sistem yang dibutuhkan, seperti LPD, NFS dan DNS.

