Praktikum

9

Manajemen Proses

A. T U J U A N

1. Mengenal konsep proses di Linux
2. Mengerti konsep sinyal dan bagaimana cara mengelola sinyal tersebut

B. DASAR TEORI

KONSEP PROSES DI LINUX

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**

  Proses yang diciptakan oleh pemakai langsung pada terminal (interaktif, dialog)
- **Batch**

  Proses yang dikumpulkan dan dijalankan secara sekuensial (satu persatu). Prose Batch tidak diasosiasikan (berinteraksi) dengan terminal.
- **Daemon**

  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

SINYAL

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

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

Nomor sinyal : 1 s/d maksimum nomor sinyal yang didefinisikan system

Standar nomor sinyal yang terpenting adalah :

No Sinyal	Nama	Deskripsi
1	SIGHUP	Hangup, sinyal dikirim bila proses terputus, misalnya melalui putusnya hubungan modem
2	SIGINT	Sinyal interrupt, melalui ^C
3	SIGQUIT	Sinyal Quit, melalui ^\
9	SIGKILL	Sinyal Kill, menghentikan proses
15	SIGTERM	Sinyal terminasi software

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
```

Sebelum mengirim sinyal PID proses yang akan dikirim harus diketahui terlebih dahulu.

C. LANGKAH – LANGKAH

1. Masuk ke sistem operasi Linux.
2. Login sebagai `stD3XXYYY`.
3. Gunakan instruksi status proses : `ps`.
4. Gunakan instruksi untuk mengelola sinyal : `kill, trap, nohup`

D. PERCOBAAN

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
   ```

2. Untuk melihat factor/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
   ```

3. Mencari proses yang spesifik pemakai. Proses diatas hanya terbatas pada proses milik pemakai, dimana pemakai teresbut melakukan login

   ```
   $ ps –u stD3XXYYY
   ```

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

   ```
   $ ps –a
   $ ps –au
   ```

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
   ```

2. Eksekusi file loop.sh sebagai background

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

3. Melihat proses id

```
$ ps
```

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

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

5. Menghentikan proses secara mutlak

```
$ kill –9 [nomor PID]
```

Percobaan 3 : Mengelola sinyal

1. Membuat file prog.sh

```
$ vi prog.sh
#!/bin/sh
echo "Program berjalan …"
while :
do
  echo "X"
  sleep 20
done
```

2. Jalankan program tersebut. Karena program melakukan looping, maka stop dengan mengirim sinyal interrupt (^C)

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

4. Kirimkan sinyal terminasi sebagai berikut

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

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
```

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]
```

7. Perintah kill diatas tidak akan menghentikan proses karena dihalangi dengan perintah trap. Cobalah menggunakan Nomor sinyal 9

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

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 sort

```
$ vi myjob.sh
#!/bin/sh
i=1
while :
do
  find / -print > berkas
  sort berkas -o hasil
  echo "Proses selesai pada 'date'" >> proses.log
  sleep 60
done
```

2. Jalankan proses tersebut sebagai proses background

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

3. Kemudian logout dan login kembali. Periksa sampai dimana job bekerja.

```
$ ps
```

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
```

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

```
$ ps
```

E. Latihan

1. Login sebagai `studentOS` dan lihat status proses, perhatikan kolom keluaran ps –au sebagai berikut :
   a. Sebutkan nama-nama proses yang bukan root
   b. Tulis PID dan COMMAND dari proses yang paling banyak menggunakan CPU time
   c. Sebutkan buyut proses dan PID dari proses tersebut
   d. Sebutkan beberapa proses daemon
   e. Pada prompt login lakukan hal-hal sebagai berikut :

      ```
      $ csh
      $ who
      $ bash
      $ ls
      $ sh
      $ ps
      ```

      Sebutkan PID yang paling besar dan kemudian buat urut-urutan proses sampai ke PPID = 1. Lakukan ^d atau exit atau logout sampai kembali muncul login: prompt
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
```

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 diatas ?

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 !