# BAB 8 Administrasi Sistem dalam Mode Teks pada Sistem Operasi Jaringan

**Tujuan:**

Pembahasan ini bertujuan agar siswa dapat :
1. bekerja dengan Command line Interface
2. Menangani File dan Direktori
3. Menjelaskan Pipeline dan Redirection
4. Melakukan Administrasi Sistem
5. Melakukan Backup Data

**Pokok Bahasan**

Dalam pembahasan ini meliputi:
1. Command Line Interface
2. Penanganan file dan Direktori pada shell
3. Pipeline dan Redirection
4. Administrasi Sistem
5. Backup data

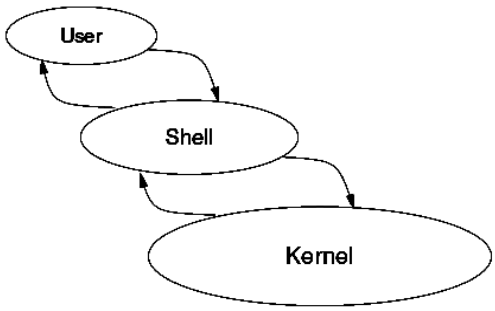
Sebuah sistem komputer, tetap membutuhkan perhatian dari manusia supaya dapat berjalan dengan baik. Mungkin sistem tersebut tetap dapat berjalan tanpa harus mendapatkan perhatian penuh dari manusia, namun jika terjadi masalah pada sistem tersebut, peranan manusia tetap diperlukan. Manusia yang bertugas untuk menangani hal ini disebut *administrator sistem* atau *system administrator*, disingkat menjadi *sysadmin* atau *admin* saja.

Sebuah sistem komputer tidak akan dapat bekerja dengan baik jika salah satu dari ketiga komponen ini dihilangkan, yaitu: *hardware, software, brainware*. Hardware adalah bagian komputer yang tampak secara fisik. Software adalah kode-kode instruksi yang dijalankan pada hardware yang bersangkutan. Sedangkan brainware adalah manusia yang bertugas untuk

mengoperasikan sistem komputer. Administrasi sistem adalah aspek yang berkaitan erat pada faktor brainware tersebut. Dalam bagian ini akan dibahas mengenai administrator sistem dan tugas-tugas apa saja yang harus dilakukannya. Sistem operasi yang digunakan adalah sistem operasi Linux.

## 8.1. Bekerja dengan Command Line Interface

## 8.1. Shell




**Gambar 8 - 1 User berkomunikasi dengan kernel melalui shell**

*Shell*, dalam komputer adalah salah satu jenis program bawaan sistem operasi (seringnya merupakan program yang terpisah dari inti sistem operasi) yang menyediakan komunikasi langsung antara pengguna dan sistem operasi. Begitu juga dalam linux shell merupakan interface atau antar muka

yang menghubungkan antara user dengan kernel (Lihat gambar 8.1).

Perlu diingatkan bahwa shell tidak selalu berupa interface berbentuk teks tapi juga dapat berbentuk interface grafik. Shell tradisional dari linux adalah berupa teks atau sering disebut juga *command line interface*.
Macam-macam shell pada linux:

- The Bourne shell (sh)
  sh adalah shell standar Unix yang dibuat tahun 1979 oleh Stephen Bourne dari AT&T dengan memakai bahasa pemrograman Algol. sh terkenal karena sederhana, compact, and cepat. Kelemahannya adalah kurang interkatif seperti tidak ada history, aliasing, dan job control. Default prompt shell sh adalah $ (dolar)

- C shell (csh)
  csh memiliki fitur yang lebih lengkap dibandingkan sh. Shel ini dibuat tahun 1970 oleh Bill Joy dari University of California at Berkeley dengan menggunakan bahasa C.
  Fitur yang terdapat dalam csh antara lain command-line history, aliasing, built-in arithmetic, filename completion, dan job control. Kelemahnnya adalah karena didesain untuk mesin skala besar dan memiliki banyak fitur maka shel ini cenderung lambat bila digunakan pada mesin kecil
  Default prompt shell csh adalah % (persen)

- Korn Shell (ksh)
  Korn shell merupakan pengembangan dari bourne shell yang ditulis oleh *David Korn* dari AT&T pada pertengahan 1980an. fitur Korn shell antara lain editable history, aliases, functions, regular expression wildcards, built-in arithmetic, job control, coprocessing, dan special debugging.
  Default prompt shell ksh adalah $ (dolar)

- The GNU Bourne Again shell (bash)
  Bash merupakan default shell Linux yang merupakan pengembangan dari bourne shell sehingga kompatibel juga di Unix. Shell ini dibuat pada tahun 1988 oleh Brian Fox dari FSF GNU. Fitur yang dimiliki bash antara lain interaktif, dapat membuat shortcut, bisa berwarna, dll. Default Bash prompt adalah $ (dolar)

- TC shell (tcsh)
  TC shell merupakan prominent shell untuk Linux yang kompatibel juga di Unix. TC shell compatible dengan csh nya unix dan memiliki fitur yang paling lengkap. Oleh karena itulah shel ini menjadi shell faforitku. Fitu tersebut antara lain command-line editing (emacs dan vi), scrolling the history list, advanced filename, variable, and command completion, spelling correction, job scheduling, automatic locking and logout, time stamps in the history list, dll.
  Default C shell prompt adalah > (the greater-than sign)

- Z shell (zsh)
  zsh berusaha menggabungkan firtur dari bash, tcsh, dan ksh.

### 8.2.2. Bash

*Bourne Again Shell* (*Bash*) dibuat untuk digunakan dalam proyek GNU. Proyek

GNU dimulai oleh *Richard M Stallman* untuk membuat sistem operasi yang kompatibel dengan UNIX dan menggantikan seluruh utilitas UNIX komersil dengan yang bebas didistribusikan.

*Bash*, yang ditujukan menjadi shell standar sistem GNU, secara resmi "lahir" pada hari Minggu, 10 Januari 1998. *Brian Fox* menulis versi awal *bash* dan *readline* dan terus memperbaikinya hingga tahun 1993. Di awal tahun 1989, *Chet Ramey* bergabung, dan kini ia adalah pemelihara resmi *bash* dan terus membuat peningkatan-peningkatan lebih lanjut.

*Bash* kini semakin populer, karena ia umumnya disertakan dalam setiap sistem operasi UNIX. Sebagai tambahan pada kompatibilitasnya dengan *shell Bourne, Bash* juga menyertakan fitur-fitur terbaik *shell C* dan *Korn*, dan juga beberapa fitur unik yang dimilikinya.

Mode edit pada perintah baris *bash* adalah sebuah fitur yang cenderung menarik orang untuk menggunakannya. Dengan edit perintah baris, lebih mudah untuk kembali dan membetulkan kesalahan atau memodifikasi perintah-perintah sebelumnya.

Fitur utama *bash* lainnya adalah ditujukan bagi user interaktif yaitu kendali perintah (*job control*). Kendali perintah memberikan anda kemampuan untuk memulai, menghentikan dan berhenti sejenak (*pause*) sejumlah perintah di waktu yang bersamaan. Fitur ini dipinjam dari *shell C*.

Keunggulan-keunggulan penting lain *bash* umumnya ditujukan bagi programmer atau orang yang hobi mengkustomisasi *shell*. *Bash* menyertakan banyak pilihan dan variabel baru untuk kustomisasi, dan fitur pemrogramannya telah sangat diperbaiki untuk menyertakan definisi fungsi, lebih banyak kendali struktur, aritmatika integer, kendali I/O tingkat tinggi, dan lain-lain.

Untuk mencari tahu *shell* yang sedang digunakan, dapat diberikan perintah berikut:

```
echo $SHELL
```

Jika belum menggunakan *bash* dan ingin menggunakannya, pertama-tama harus diketahui apakah sudah terdapat bash pada sistem. Ketikkan *bash*. Jika diperoleh prompt dollar ($), maka *bash* telah terpasang pada sistem, ketikkan *exit* untuk kembali ke *shell* normal.

Jika didapatkan pesan *"not found"*, kemungkinan *bash* belum terpasang pada sistem. Untuk memasangnya diperlukan hak akses sebagai *administrator* atau *root*.

Untuk menjadikan *bash* sebagai *shell* normal (*default*), berikan perintah berikut:

```
$ chsh <lokasi_bash>
```

*lokasi_bash* bisa dicari dengan menggunakan perintah *which bash*. Jika berhasil, pada login berikutnya shell normalnya adalah *bash*.

### 8.2.3. Perintah, Argumen, dan Option

Perintah baris shell terdiri dari satu atau lebih kata, yang dipisahkan oleh blank atau TAB. Format periintahnya adalah sebagai berikut:

```
perintah [option][argument]
```

Kata pertama pada baris adalah perintah. Sisanya (bila ada) adalah argumen (juga disebut parameter) bagi perintah.

Sebuah option adalah tipe argumen khusus yang memberikan perintah informasi khusus mengenai apa yang harus dilakukannya. Option biasanya terdiri dari sebuah tanda "-" yang diikuti oleh sebuah huruf. Perintah *lp -h myfile* mengandung option *-h*, yang memberitahu printer untuk tidak mencetak halaman banner sebelum mencetaknya ke file.

### 8.2.4. Nama file, wildcard, dan ekspansi path

Terkadang diperlukan untuk menjalankan sebuah perintah pada lebih dari satu file. Contoh paling umum adalah perintah *ls*, yang menampilkan informasi tentang file. Dalam bentuk paling sederhananya, tanpa option atau argumen. *ls* menampilkan nama semua file dalam direktori kerja kecuali file-file tersembunyi, yang namanya dimulai dengan tanda titik (.).

Nama file begitu penting dalam UNIX sehingga shell menyediakan cara built-in untuk menspesifikasikan pola sejumlah nama file tanpa perlu tahu nama mereka. Anda dapat menggunakan karakter-karakter khusus, yang disebut *wildcard*, dalam nama file untuk menjadikan mereka pola. Tabel 8-1 menampilkan sejumlah *wildcard* dasar:

**Tabel 8 - 1 Daftar Wildcard**

| Wildcard | Kecocokan |
|---|---|
| ? | Sembarang karakter tunggal |
| * | Sembarang string karakter |
| [set] | Sembarang karakter yang ada di set |
| [!set] | Sembarang karakter yang tidak ada di set |

*Wildcard ?* akan mencocokkan satu buah karakter, jadi bila direktori anda berisikan file-file *program.c*, *program.log,* dan *program.o,* maka ekspresi *program.?* akan cocok dengan *program.c* dan *program.o* namun tidak dengan *program.log*.

Tanda asteriks (*) lebih powerful dan lebih banyak digunakan. Ia akan mencocokkan sembarang string karakter. Ekspresi *program.** akan cocok dengan ketiga file tersebut di atas. Perhatikan bahwa tanda (*) dapat juga cocok dengan string kosong, **ed* dan **e** akan cocok dengan *ed*.

*Wildcard* sisanya adalah konstruksi *set*. Sebuah *set* adalah sebuah daftar karakter (misalnya *abc*), sebuah rentang inklusif (misalnya *A-Z*), atau keduanya.

Dalam contoh di atas, *program.[co]* dan *program.[a-z]* akan cocok dengan *program.c* dan *program.o*, namun tidak *program.log*.

Tanda "seru" setelah kurung siku kiri menjadikan sebuah set *ternegasi*. Sebagai contoh [*!a-zA-Z*] akan cocok dengan sembarang karakter yang bukan huruf seperti angka (0-9) dan simbol (!,@,#,$,%,^,&,*,dll).

### 8.2.5. Kustomisasi Lingkungan Shell

Sinonim untuk shell UNIX atau untuk interface sembarang program komputer adalah sebuah *environment*. *Environment* biasanya merupakan kumpulan konsep yang menyatakan hal-hal yang dilakukan komputer dalam bidang yang dapat dimengerti dan

koheren, dan sebuah tampilan dan rasa yang nyaman.

Sebagai contoh, meja kerja adalah sebuah environment. Konsep-konsep yang ada dalam meja kerja biasanya adalah memo, panggilan telpon, surat, formulir, dsb. Tool yang biasa ada adalah kertas, staples, amplop, pen, telpon, kalkulator, dsb. Setiap benda ini memiliki karakteristik yang menyatakan bagaimana menggunakan mereka. Karateristik tersebut berkisar dari lokasi pada meja hingga ke hal yang lebih canggih seperti nomor apa yang disimpan pada tombol memori telpon.

Kustomisasi tampilan dan rasa lingkungan kerja dilakukan dengan menaruh pulpen di tempat yang mudah dijangkau, memprogram tombol telpon, dsb. Secara umum, semakin banyak kustomisasi yang dilakukan, semakin sesuai dengan kebutuhan pribadi, dan karenanya menjadi lebih produktif.

UNIX shell juga memiliki karakteristik yang hampir sama dengan meja kerja. Namun tampilan dan rasa lingkungan UNIX ditentukan dengan keyboard dan tampilan, juga dengan cara bagaimana mensetup direktori, di manatempat menaruh file, dan nama apa saja yang digunakan untuk file, direktori, dan perintah.

*Bash* menyediakan beberapa fitur untuk mengkustomisasi lingkungan kerja *bash*:

- File khusus, seperti *".bash_profile", ".bash_logout"*, dan *".bashrc"* yang dibaca oleh *bash* ketika anda login atau memulai shell baru.
- *Alias*. Sinonim untuk perintah atau string perintah yang dapat didefinisikan untuk kemudahan.
- *Option*. Mengendalikan berbagai aspek lingkungan shell.
- *Variabel*. Nilai-nilai yang dapat diubah yang diacu dengan sebuah nama.

**Tabel 8 - 2 File kustomisasi lingkungan kerja bash**

| Nama File | Penjelasan |
|---|---|
| ~/.bash_login | Dipakai oleh bash dan bila tidak ada file ~/.bash_profile yang dipakai |
| ~/.bash_logout | Dipakai oleh bash ketika keluar dari program bash |
| ~/.bash_profile | Dipakai oleh bash saat login setelah /etc/profile. |
| ~/.bash_history | Daftar program yang telah dijalankan sebelumnya |
| ~/.bashrc | Dipakai oleh bash untuk mendefinisikan variabel-variabel |
| ~/.vimrc | File konfigurasi default program vim |
| ~/.gtkrc | GNOME Toolkit. |
| ~/.kderc | Konfigurasi KDE. |
| ~/.rhosts | Digunakan oleh program r-tools: rsh, rlogin, dsb. Sangat tidak aman karena mudah untuk memasuki sistem. 1. Harus dimiliki oleh user atau superuser. 2. Menampilkan host tempat user dapat mengakses account ini 3. Diabaikan jika berupa link simbolik |
| ~/.xinitrc | Dibaca oleh progam X saat dijalankan. Kebanyakan menjalankan program tertentu, misalnya exec gnome-session Bila perintah di atas ada dalam file ini berarti pada saat dijalankan, GNOME akan dijalankan pula saat user menjalankan program startx |

## File.*bash_profile*,.*bash_logout*, dan.*bashrc*

Ketiga file ini yang ada pada direktori *home* memiliki arti khusus bagi *bash*, yaitu menyediakan sebuah cara bagi pengguna untuk mensetup lingkungan user secara otomatis ketika user login dan ketika user memulai shell bash lain, dan memungkinkan user melakukan perintah-perintah ketika user logout. File bash yang paling penting adalah *.bash_profile*, yang dibaca dan perintah-perintah yang ada di dalamnya dieksekusi oleh bash setiap kali user login ke sistem.

Berikut ini adalah contoh file *.bash_profile*:

```
#.bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
      . ~/.bashrc
fi

export BASH_ENV=$HOME/.bashrc
```

Baris-baris ini mendefinisikan lingkungan dasar untuk akun login user. Ketika mengedit *.bash_profile* user, cukup tambahkan baris-baris baru user di bagian akhir file.

Apapun yang user tambahkan ke *.bash_profile* tidak akan memiliki pengaruh hingga file tersebut dibaca kembali dengan cara logout dan login kembali. Selain itu user juga dapat menggunakan perintah *source*. Sebagai contoh:

```
source.bash_profile
```

Bash membolehkan dua buah sinonim untuk *.bash_profile* yaitu.*bash_login* dan *.profile*. Jika *.bash_profile* tidak ada dalam home direktori user maka bash akan mencari.*bash_login*. Jika tidak ada juga, ia akan mencari *.profile*.

*.bash_profile* dibaca dan dieksekusi hanya oleh shell login. Jika user memulai shell baru dengan mengetikkan *bash* pada perintah baris, ia akan membaca perintah-perintah dari file *.bashrc*.

Berikut ini adalah contoh file.*bashrc*:

```
#.bashrc

# Source global definitions
if [ -r /etc/bashrc ]; then
      . /etc/bashrc
fi
```

*.bash_logout* dibaca dan dieksekusi setiap shell login keluar. Jika user mengeksekusi beberapa perintah yang menghapus file-file temporer dari akun user atau mencatat berapa lama user telah login ke sistem maka user dapat menempatkan perintah-perintah dalam.*bash_logout*.

Berikut ini adalah contoh file.*bash_logout*:

```
# ~/.bash_logout

Clear
```

### *Aliases*

Jika user telah cukup lama menggunakan perintah-perintah UNIX mungkin user tahu terdapat banyak perintah dan beberapa dari mereka memiliki nama yang aneh. Terkadang perintah yang sering digunakan memiliki sejumlah option dan argumen yang perlu dimasukkan. Bash memiliki sebuah fitur yang memungkinkan user mengganti nama perintah atau memungkinkan user mengetikkan sesuatu yang mudah untuk menggantikan banyak pilihan. Fitur tersebut adalah *alias*.

*Alias* dapat didefinisikan pada perintah baris, dari dalam *.bash_profile* atau dalam *.bashrc* user, dengan menggunakan bentuk:

```
alias nama=perintah
```

Dengan *nama* adalah sebuah alias bagi perintah. Jika user mengetikkan *nama* sebagai perintah, bash akan menggantikan perintah ketika ia mengeksekusinya. Perhatikan bahwa tidak ada spasi pada kedua sisi tanda *sama dengan* (=).

Contoh:

```
alias dir='ls -F'
```

Jika user memberikan perintah *dir* maka akan dieksekusi *ls* dengan option *-F*.

Berikut ini adalah hal mengenai *alias* yang penting untuk diingat:

- bash membuat penggantian tekstual terhadap alias. Bayangkan bash memberikan perintah user ke sebuah teks editor dan melakukan "ubah" atau "ganti" sebelum menginterpretasi dan mengeksekusinya.
- alias adalah rekursif, yang berarti mungkin untuk mengaliaskan sebuah alias.
- alias hanya dapat digunakan pada bagian awal string perintah.

- jika user memberikan perintah alias nama tanpa tanda *sama dengan* (=) dan nilainya, shell akan mencetak *nilai alias* atau *alias name not found* jika tidak didefinisikan. Jika user mengetikkan alias tanpa argumen, akan ditampilkan daftar semua alias yang telah didefinisikan. Perintah *unalias nama* akan menghapus definisi *alias nama*.

***Option***

Meskipun alias membolehkan user membuat nama yang nyaman bagi perintah-perintah, mereka tidak mengubah perilaku shell. Untuk mengubahnya dapat digunakan *option*. *Option* shell adalah sebuah seting yang berada dalam keadaan "on" atau "off".

**Tabel 8 - 3 Daftar Opsi**

| Option | Deskripsi |
|---|---|
| emacs | Memasuki mode edit emacs (bakunya on) |
| Ignoreoff | Jangan bolehkan penggunaan CTRL-D tunggal untuk log off; gunakan perintah exit untuk log off dengan segera. |
| noclobber | Jangan bolehkan redireksi output ( > ) untuk menimpa file yang telah ada. |
| Noglob | Jangan ekspansikan nama file wildcard seperti * dan ?. |
| Nounset | Mengindikasikan kesalahan ketika berusaha menggunakan sebuah variabel yang belum didefinisikan. |
| Vi | Memasuki mode edit vi. |

Perintah dasar yang berkaitan dengan option adalah set *-o nama_option* dan set *+o nama_option*. Tanda - menyebabkan sebuah option *"on"*, sementara tanda + membuat sebuah option menjadi *"off"*.

Tabel 8-3 menampilkan *option-option* yang berguna bagi user UNIX umum. Semuanya secara baku diset menjadi off:

### 8.2.6. Variabel Shell

Terdapat berbagai karakteristik lingkungan user yang mungkin ingin dikustomisasi namun tidak dapat dilakukan dengan pilihan *on/off*. Karakteristik tipe ini dispesifikasikan dalam *variabel shell*.

Seperti *alias*, sebuah *variabel shell* adalah sebuah nama yang memiliki nilai terasosiasi dengannya. Bash menyediakan beberapa buah *variabel shell built-in*, programer shell dapat menambahkan variabel sendiri. Secara konvensi, *variabel-variabel built-in* memiliki nama dalam huruf kapital. *Variabel built-in* yang bash-spesifik, didefinisikan dalam huruf kecil.

Format untuk mendefinisikan *variabel* serupa dengan penulisan *alias*:

```
varname=value
```

Tidak boleh ada spasi di kedua belah sisi *tanda sama dengan*, dan jika nilai lebih dari satu kata, ia harus diapit oleh tanda ("). Untuk menggunakan nilai sebuah *variabel* dalam sebuah perintah, berilah tanda dolar ($) di awal nama variabel.

Contoh untuk setting dan memanggil nilai dari variabel:

```
tekaje@ns:~$ set TKJ
tekaje@ns:~$ TKJ=qwerty
tekaje@ns:~$ echo $TKJ
qwerty
tekaje@ns:~$
```

Untuk menghapus sebuah *variabel* gunakan perintah:

```
$ unset nama_variabel
```

Untuk memeriksa nilai sebuah *variabel* dapat digunakan perintah *echo*.

## 8.2. Penanganan File dan Direktori pada Shell

### 8.2.1. Navigasi: ls, cd, dan pwd

#### ls

Perintah ini menampilkan daftar file pada sebuah direktori. Pengguna Windows dan DOS akan menemukan kesamaan pada perintah *dir*. Jika dijalankan sendiri, *ls(1)* akan menampilkan file pada direktori yang aktif. Untuk melihat apa yang ada pada direktori *root*, anda bisa menjalankan perintah ini:

```
$ cd /
$ ls
```

Maka akan ditampilkan seperti dibawah ini.

```
bin cdr dev home lost+found proc
sbin tmp var boot cdrom etc lib
mnt root suncd usr vmlinuz
```

Masalah yang banyak dialami pengguna dengan hasil keluaran adalah sulit membedakan antar direktori dan file. Beberapa pengguna lebih memilih *ls* dengan menambahkan penanda jenis pada setiap daftar, seperti dibawah ini:

```
$ ls –FC
bin/ cdr/ dev/ home/ lost+found/
proc/ sbin/ tmp/ var/ boot/
cdrom/ etc/ lib/ mnt/  root/
suncd/ usr/ vmlinuz
```

Direktori akan mendapatkan sebuah tanda slash diakhir namanya, file yang dapat dieksekusi akan mendapatkan sebuah tanda bintang (asterisk), dan seterusnya.

*ls* juga bisa digunakan untuk mendapatkan statistik lain pada file-file. Sebagai contoh, untuk melihat tanggal pembuatan, kepemilikan, dan hak akses, dapat ditampilkan daftar yang lebih panjang:

```
$ ls -l
drwxr-xr-x 2 root bin 4096 May 7 09:11 bin/
drwxr-xr-x 2 root root 4096 Feb 24 03:55 boot/
drwxr-xr-x 2 root root 4096 Feb 18 01:10 cdr/
drwxr-xr-x 14 root root 6144 Oct 23 18:37 cdrom/
drwxr-xr-x 4 root root 28672 Mar 5 18:01 dev/
drwxr-xr-x 10 root root 4096 Mar 8 03:32 etc/
drwxr-xr-x 8 root root 4096 Mar 8 03:31 home/
drwxr-xr-x 3 root root 4096 Jan 23 21:29 lib/
drwxr-xr-x 2 root root 16384 Nov 1 08:53 lost+found/
drwxr-xr-x 2 root root 4096 Oct 6 12:47 mnt/
dr-xr-xr-x 62 root root 0 Mar 4 15:32 proc/
drwxr-x--x 12 root root 4096 Feb 26 02:06 root/
drwxr-xr-x 2 root bin 4096 Feb 17 02:02 sbin/
drwxr-xr-x 5 root root 2048 Oct 25 10:51 suncd/
drwxrwxrwt 4 root root 487424 Mar 7 20:42 tmp/
```

```
drwxr-xr-x 21 root root 4096 Aug 24 03:04 usr/
drwxr-xr-x 18 root root 4096 Mar 8 03:32 var/
```

Misalkan Anda hendak melihat daftar lengkap dari file-file tersembunyi pada direktori yang aktif. Perintah ini akan melakukannya:

```
$ ls -a
. bin cdrom home mnt sbin usr
.. boot dev lib proc suncd var
.pwrchute_tmp cdr etclost+found root tmpvmlinuz
```

File-file yang diawali dengan tanda titik (disebut file bertitik - dot files) bersifat tersembunyi ketika Anda menjalankan *ls*. Anda hanya akan melihat mereka jika Anda memberikan opsi *-a*.

Terdapat lebih banyak opsi yang dapat ditemukan pada halaman manual online. Jangan lupa bahwa Anda bisa mengkombinasikan opsi-opsi yang dapat Anda berikan pada *ls*.

## cd

Perintah cd digunakan untuk mengganti direktori kerja. Anda cukup mengetikkan *cd* diikuti dengan nama path baru. Berikut ini beberapa contoh:

```
server:~$ cd /bin
server:/bin$ cd usr
bash: cd: usr: No such file or directory
server:/bin$ cd /usr
server:/usr$ ls
bin
server:/usr$ cd bin
server:/usr/bin$
```

Perhatikan bahwa tanpa tanda slash diawal, maka cd akan mencoba berpindah ke sebuah direktori pada direktori aktual. Mengeksekusi *cd* tanpa opsi akan memindahkan Anda ke direktori home.

Perintah *cd* tidak seperti perintah lain. Perintah ini adalah perintah built-in pada shell yang telah dibahas. Tidak ada halaman manual untuk perintah ini. Anda harus menggunakan bantual shell. Seperti ini:

```
$ help cd
```

Hal ini akan menampilkan opsi untuk *cd* dan bagaimana menggunakannya.

## pwd

Perintah *pwd* digunakan untuk menampilkan lokasi aktual Anda. Untuk menggunakan perintah *pwd* cukup ketikkan *pwd*. Sebagai contoh:

```
$ cd /bin
$ pwd
/bin
$ cd /usr
$ cd bin
$ pwd
```

```
/usr/bin
```

### 8.2.2. Pagers: more, less, dan most

#### more

more(1) adalah perintah yang kita sebut utilitas pager. Seringkali hasil keluaran dari sebuah perintah terlalu panjang untuk satu layar. Perintah individu tidak tahu bagaimana menyesuaikan hasil keluarannya pada layar yang terpisah. Mereka memberikan tugas ini pada utilitas pager.

Perintah more memecah hasil keluaran pada layar-layar individu dan menunggu anda menekan tombol spasi (space bar) sebelum melanjutkan ke layar selanjutnya. Menekan tombol enter akan menampilkan satu baris berikutnya. Berikut adalah contoh yang baik:

```
$ cd /usr/bin
$ ls -l
```

Hal tersebut akan menggulung hasil perintah dalam waktu yang singkat sehingga jika baris yang dihasilkan sangat banyak maka yang akan terlihat hanya bagian terakhir saja. Untuk memecah layar keluaran per layar, gunakan pipe melalui more:

```
$ ls -l | more
```

Karakter shift backslash adalah karakter pipe (). Pipe adalah singkatan untuk mengatakan ambil hasil keluaran dari *ls* dan berikan kepada *more*. Anda bisa mem-pipe semua perintah melalui more, tidak hanya ls.

#### less

Perintah *more* cukup bermanfaat, tetapi seringkali anda akan menemukan bahwa anda telah melewati layar yang anda inginkan. Perintah *more* tidak menyediakan sebuah cara untuk kembali. Perintah *less(1)* menyediakan fungsionalitas ini. Perintah ini digunakan dengan cara yang sama dengan perintah *more*, sehingga contoh sebelumnya juga berlaku disini. Jadi, *less* lebih dari *more*.

Arti *less* adalah lebih sedikit, tetapi hasilnya lebih banyak daripada *more*, sehingga *more* lebih sedikit dari *less*.. Jadi, gunakan lebih sedikit (*less*) jika Anda hendak memperoleh lebih banyak (*more*).

#### most

Pada saat *more* dan *less* tidak mampu, *most(1)* mengambil alih. Jika *less* lebih banyak dari *more*, *most* lebih banyak dari *less*. Ketika perintah pager lain hanya hampu menampilkan satu file pada satu saat yang sama, *most* mampu melihat beberapa file, sepanjang jendela setiap file paling sedikit dua baris. *most* memiliki banyak opsi, lihat halaman manual untuk detail selengkapnya. Contoh penggunaan perintah most adalah:

```
$ ls -l | most
```

Atau:

```
$ most [file_teks]
```

### 8.2.3. Keluaran Sederhana: cat dan echo

#### cat

*cat(1)* kependekan dari “concatenate”, artinya “menggabung”. Pada awalnya didesain untuk menggabungkan file teks menjadi satu, tetapi dapat digunakan untuk tujuan lainnya.

Untuk menggabungkan dua file atau lebih menjadi satu, anda cukup mendaftar file-file setelah perintah *cat* dan mengalihkan hasil keluaran pada sebuah file. *cat* bekerja dengan hasil masukan dan keluaran standar, sehingga anda harus menggunakan karakter pengalihan shell. Sebagai contoh:

```
$ cat file1 file2 file3 > bigfile
```

Perintah ini mengambil isi dari *file1, file2*, dan *file3* dan menggabungkannya. Hasil keluaran baru dikirimkan ke keluaran standar.

Seseorang dapat juga menggunakan cat untuk menampilkan file. Banyak orang melakukan *cat* terhadap file teks melalui *more* atau *less*, seperti berikut:

```
$ cat file1 | more
```

Perintah tersebut akan menampilkan file *file1* dan mem-pipe melalui *more* sehingga anda hanya mendapatkan satu layar pada satu waktu.

Penggunaan lain dari *cat* adalah untuk menyalin file. Anda bisa menyalin sembarang file dengan *cat*, seperti berikut:

```
$ cat /bin/bash > ~/mybash
```

Program */bin/bash* disalin pada direktori home anda dan diberi nama *mybash*.

*cat* memiliki banyak kegunaan dan yang dibahas disini hanya sebagian. Karena cat banyak menggunakan masukan dan keluaran standar, maka sangatlah ideal untuk digunakan pada skrip shell atau sebagai bagian dari perintah kompleks lainnya.

#### echo

Perintah *echo(1)* menampilkan teks yang spesifik pada layar. anda menentukan string yang akan ditampilkan setelah perintah *echo*. Secara default *echo* akan menampilkan string dan menampilkan karakter untuk pindah baris setelahnya. Anda bisa memberikan opsi *-n* untuk mengurangi pencetakan baris baru. Opsi *-e* akan mengakibatkan echo untuk mencari karakter escape pada string dan mengeksekusinya. Contoh pilihan –n akan membuat echo tidak menghasilkan baris baru:

```
tekaje@ns:/$ echo -n Selamat Pagi

Selamat Pagi

tekaje@ns:/$
```

### 8.2.4. Pembuatan: touch dan mkdir

#### touch

*touch(1)* digunakan untuk mengubah *timestamp* pada sebuah file. Anda bisa mengubah *timestamp* akses dan *timestamp* modifikasi dengan perintah ini. Jika file yang disebutkan tidak ada, *touch* akan membuat sebuah file kosong dengan nama yang disebutkan. Untuk menandai sebuah file dengan waktu sistem aktual, Anda bisa menggunakan perintah ini:

```
$ ls -al file1
-rw-r--r-- 1 root root 9779 Feb 7 21:41 file1
$ touch file1
$ ls -al file1
-rw-r--r-- 1 root root 9779 Feb 8 09:17 file1
```

Terdapat beberapa opsi untuk *touch*, termasuk opsi untuk menentukan *timestamp* mana yang akan dimodifikasi, waktu yang digunakan, dan masih banyak lagi. Halaman manual menjelaskan hal ini lebih detail.

**mkdir**

*mkdir(1)* akan menciptakan sebuah direktori baru. Anda bisa menentukan direktori yang akan dibuat ketika anda menjalankan *mkdir*. Contoh berikut membuat direktori *jarkom* pada direktori aktual:

```
$ mkdir jarkom
```

Anda juga bisa menentukan sebuah path, seperti ini:

```
$ mkdir /usr/local/jarkom
```

Opsi *-p* akan memberitahu *mkdir* untuk membuat direktori induk. Contoh diatas akan gagal jika /usr/local tidak ada. Opsi *-p* akan membuat /usr/local dan /usr/local/jarkom:

```
$ mkdir -p /usr/local/jarkom
```

### 8.2.5. Menyalin dan Memindahkan

**cp**

*cp(1)* menyalin file-file. Pengguna DOS akan menemukan kesamaan dengan perintah *copy*. Terdapat banyak opsi untuk *cp*, sehingga Anda harus melihat pada halaman manual sebelum menggunakannya.

Penggunaan umum dari cp adalah menyalin sebuah file dari satu lokasi ke lokasi yang lain. Sebagai contoh:

```
$ cp jarkom /tmp
```

Perintah ini menyalin file *jarkom* dari direktori aktual pada direktori */tmp*.

Banyak pengguna memilih untuk mempertahankan *timestamp*, seperti pada contoh ini:

```
$ cp -a jarkom /tmp
```

Hal ini memastikan bahwa *timestamp* tidak dimodifikasi pada saat penyalinan.

Untuk menyalin isi sebuah direktori ke direktori lain secara rekursif, gunakan:

```
$ cp -R mydir /tmp
```

Hal ini akan menyalin direktori *mydir* ke direktori */tmp*.

Juga, jika Anda hendak menyalin sebuah direktori atau file dan mempertahankan hak akses dan *timestamp*, gunakan *cp -p*.

```
$ ls -l file
-rw-r--r-- 1 root vlad 4 Jan 1 15:27 file
```

```
$ cp -p file /tmp
$ ls -l /tmp/file
-rw-r--r-- 1 root vlad 4 Jan 1 15:27 file
```

**mv**

*mv(1)* memindahkan satu file ke tempat lain. Terkesan mudah bukan?

```
$ mv oldfile /tmp/newfile
```

*mv* memiliki beberapa opsi perintah baris yang dibahas secara detail pada halaman manual. Pada umumnya, *mv* hampir tidak pernah digunakan dengan opsi perintah baris.

### 8.2.6. Penghapusan: rm dan rmdir

**rm**

*rm(1)* menghapus file-file dan struktur direktori. Pengguna DOS akan menemukan kesamaan pada perintah *del* dan *deltree*. Perintah *rm* dapat sangat berbahaya jika anda tidak hati-hati. Meskipun dimungkinkan untuk mendapatkan file yang sudah dihapus, hal ini bisa rumit (dan mungkin mahal) dan diluar batasan buku ini.

Untuk menghapus sebuah file, tentukan nama file ketika anda menjalankan *rm*:

```
$ rm file1
```

Jika file tidak memiliki hak akses tulis, maka Anda mungkin mendapatkan pesan kesalahan hak akses ditolak. Untuk memaksa penghapusan sebuah file, gunakan opsi *-f*, seperti ini:

```
$ rm -f file1
```

Untuk menghapus seluruh direktori, Anda menggunakan opsi *-r* dan *-f* bersamaan. Berikut ini contoh bagus untuk menghapus seluruh isi hard disk anda. Anda tidak ingin melakukan hal ini. Tetapi berikut adalah contohnya:

```
# rm -rf /
```

Hati-hati dengan penggunaan *rm*, jangan sampai merugikan diri sendiri. Terdapat beberapa opsi perintah baris, yang dibahas secara detail pada halaman manual online.

**rmdir**

*rmdir(1)* menghapus direktori dari sistem file. Direktori harus kosong sebelum dapat dihapus. Sintaksnya:

```
$ rmdir <directory>
```

Contoh ini akan menghapus subdirektori *jarkom* pada direktori aktual:

```
$ rmdir jarkom
```

Jika direktori tidak ada, *rmdir* akan memberitahukan anda. Anda juga dapat menentukan path lengkap pada direktori yang akan dihapus, seperti contoh:

```
$ rmdir /tmp/jarkom
```

Contoh diatas juga akan mencoba menghapus direktori *jarkom* didalam direktori */tmp*.

Anda juga bisa menghapus sebuah direktori dan semua direktori induknya dengan menggunakan opsi *-p*.

```
$ rmdir -p /tmp/jarkom
```

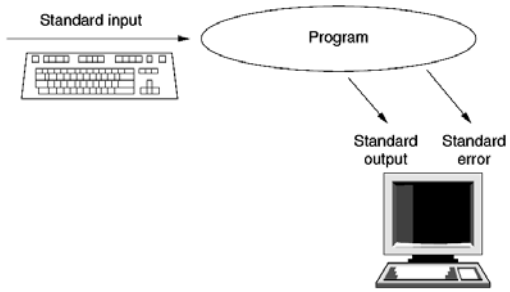
Hal ini akan mencoba menghapus direktori *jarkom* didalam */tmp* terlebih dahulu. Jika sukses, maka melanjutkan untuk menghapus */tmp*. *rmdir* akan terus berlanjut sampai sebuah kesalahan terjadi atau seluruh struktur yang diberikan dihapus.

## 8.3. Pipeline dan Redirection

### 8.4.2. Redirection

Pada sistem operasi UNIX dan Linux user bisa mengalihkan output suatu proses ke tujuan lain misalnya ke sebuah file atau layar monitor. Proses tersebut dinamakan pengalihan atau *redirection*.

Terdapat tiga terminologi yang perlu diketahui dalam proses pengalihan ini yaitu *standard input, standard output*, dan *standard error (gambar 8-2)*. *Standard input* adalah masukan dari suatu perintah atau program, defaultnya adalah keyboard.




Gambar 8.2 Terminologi input dan output

*Standard output* adalah keluaran dari suatu program, defaultnya adalah layar monitor atau terminal. Program Linux/UNIX biasanya mengeluarkan hasilnya ke *standard output*.

*Standard error* adalah keluaran dari suatu program jika terjadi error. Keluaran ini berupa pesan kesalahan yang berguna bagi pembuat program atau orang lain yang membutuhkan. *Standard error* biasanya adalah layar konsol (terminal).

Di Linux ada dua simbol yang dipakai dalam *redirection*, yaitu:

- Tanda lebih dari (>), untuk mengalihkan output ke suatu file
- Tanda kurang dari (<), untuk mengalihkan input dari suatu file

Berikut ini adalah contoh I/O *redirection*:

```
$ date > now
```

Perintah di atas akan menyimpan tanggal dan waktu sekarang ke dalam sebuah file bernama *now*.
*Redirector input* dan *output* dapat pula dikombinasikan. Sebagai contoh, perintah *cp* secara normal digunakan untuk menyalinkan file. Jika karena sesuatu alasan ia tidak ada atau rusak, anda dapat menggunakan *cat* sebagai berikut:

```
$ cat < file1 > file2
```

Hal ini serupa dengan `cp file1 file2`

### 8.4.3. Pipeline

Pada sistem operasi Linux/UNIX, hasil keluaran suatu proses program dapat dijadikan input bagi proses lainnya. Cara tersebut dikenal dengan sebutan *pipeline*.

Misalnya jika user menampilkan isi direktori */etc* dengan *ls -al*, maka hasil tampilannya akan sangat banyak dan user tidak sempat membaca nama file yang paling atas.

Sementara itu untuk user yang telah mengenal perintah *more* untuk menampilkan sesuatu layar per layar, dengan *pipeline* dapat memberikan keluaran perintah *ls -al* sebagai

masukan perintah *more*. Adapun caranya adalah sebagai berikut:

```
# ls -al /etc | more
```

Tanda vertical bar (|) adalah tanda yang digunakan untuk pipeline. Penggunaan pipeline pada perintah-perintah Linux sangat banyak.

Dengan pipeline user juga bisa menyaring hasil proses suatu program untuk ditampilkan sesuai dengan kriteria yang ditentukan. Misalnya tampilan layar per layar atau tampilan tersortir. Ada banyak perintah Linux yang dapat digunakan untuk melakukan penyaringan ini, antara lain *grep, wc, sort, cut*, dan *uniq*.

Grep digunakan untuk untuk mencetak baris yang berisi suatu string yang ditentukan oleh user. wc digunakan untuk mencetak banyaknya baris, kata, karakter, dan ukuran data dari suatu file. sort digunakan untuk melakukan penyortiran baris suatu file. cut digunakan untuk membuang bagian tertentu dari baris suatu file. uniq digunakan untuk melaporkan jika terjadi pengulangan baris.

## 8.4. Administrasi Sistem

### 8.4.1. Job Control

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 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 tombol “f” dan “g” secara serentak (”*fg*”) atau tombol “b” dan “g” secara serentak (”*bg*”). Sebagai catatan, menghentikan job sementara sangat berbeda dengan melakukan interrupt job (biasanya menggunakan [*Ctrl-C*]). Pada kasus yang terakhir job yang diinterrupt akan dimatikan secara permanen dan tidak dapat dijalankan lagi.

### 8.4.2. Membuat, Memonitor dan Mematikan Proses

Proses adalah sesuatu yang mendasar pada suatu sistem operasi. Sebuah proses dapat memboot komputer dan dapat pula mematikannya. Secara konteks sebuah proses dapat diumpamakan sebuah program yang sedang dijalankan. Proses-proses dapat dimulai/dijalankan, diakhiri/stop, dimatikan, ditentukan prioritasnya, dan dijadwalkan. Untuk memahaminya, sebagai contoh anda membuka program berarti menjalankan proses, anda menutup program berarti mematikan proses.

Dalam Linux/UNIX untuk melihat proses dan penanganan proses-proses dapat dilakukan dengan program konsole **’top’** dan **’ps’** dan untuk mematikannya

atau memanipulasinya menggunakan perintah seperti **'kill'** dan **'nice'**. Semua informasi mengenai proses yang sedang berlangsung / berjalan disimpan pada direktori **'/proc'** yang berubah secara real time. Dengan membaca file-file yang ada di direktori **'/proc'** akan sangat merepotkan, ada dua perintah yang dapat melihat proses yang sedang berlangsung.

**ps**

Perintah ps digunakan untuk melihat proses yang sedang berlangsung. Dengan menggunakan **ps** dapat dilihat informasi proses yang sedang berlangsung. Contoh tampilan **ps**

```
PID   TTY   TIME        CMD
235   tty1  00:00:00    bash
780   tty1  00:00:00    ps
```

Tampilan di atas adalah untuk menampilkan informasi proses itu sendiri. karena menjalankan **ps** di baris perintah yang menginduk pada shell command line. **ps** sendiri mempunyai banyak option namun secara umum sintaksnya adalah

```
ps [option].
```

Misalnya: **ps aux**

Jika perintah ini dijalankan, maka akan dihasilkan informasi seperti berikut ini:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3   2952  1912 ?        Ss   Feb12   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   Feb12   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   Feb12   0:01 [migration/0]
root         4  0.0  0.0      0     0 ?        SN   Feb12   0:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Feb12   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S<   Feb12   0:00 [migration/1]
root         7  0.0  0.0      0     0 ?        SN   Feb12   0:00 [ksoftirqd/1]
root         8  0.0  0.0      0     0 ?        S<   Feb12   0:00 [watchdog/1]
root         9  0.0  0.0      0     0 ?        S<   Feb12   0:00 [events/0]
root        10  0.0  0.0      0     0 ?        S<   Feb12   0:00 [events/1]
root        11  0.0  0.0      0     0 ?        S<   Feb12   0:00 [khelper]
root        31  0.0  0.0      0     0 ?        S<   Feb12   0:02 [kblockd/0]
root        32  0.0  0.0      0     0 ?        S<   Feb12   0:00 [kblockd/1]
root        33  0.0  0.0      0     0 ?        S<   Feb12   0:00 [kacpid]
root        34  0.0  0.0      0     0 ?        S<   Feb12   0:00 [kacpi_notify]
root       130  0.0  0.0      0     0 ?        S<   Feb12   0:00 [kseriod]
root       155  0.0  0.0      0     0 ?        S    Feb12   0:00 [pdflush]
root       156  0.0  0.0      0     0 ?        S    Feb12   0:01 [pdflush]
root       157  0.0  0.0      0     0 ?        S<   Feb12   0:07 [kswapd0]
root       209  0.0  0.0      0     0 ?        S<   Feb12   0:00 [aio/0]
root       210  0.0  0.0      0     0 ?        S<   Feb12   0:00 [aio/1]
root      1962  0.0  0.0      0     0 ?        S<   Feb12   0:00
[ksuspend_usbd]
root      1964  0.0  0.0      0     0 ?        S<   Feb12   0:00 [khubd]
root      2019  0.0  0.0      0     0 ?        S<   Feb12   0:00 [ata/0]
root      2024  0.0  0.0      0     0 ?        S<   Feb12   0:00 [ata/1]
root      2027  0.0  0.0      0     0 ?        S<   Feb12   0:00 [ata_aux]
root      2091  0.0  0.0      0     0 ?        S<   Feb12   0:00 [scsi_eh_0]
root      2092  0.0  0.0      0     0 ?        S<   Feb12   0:00 [scsi_eh_1]
root      2292  0.0  0.0      0     0 ?        S<   Feb12   1:40 [kjournald]
root      2452  0.0  0.1   2436   684 ?        S<s  Feb12   0:00 /sbin/udevd –
daemon
```

keterangan tentang field/kolom untuk perintah di atas

USER adalah nama user yang menjalankan proses yang bersangkutan atau atas ijin

siapa proses tersebut berjalan. Sebagian proses yang dijalankan oleh root merupakan proses boot yang dijalankan oleh *init* (central dari semua proses, dapat dilihat dengan perintah *'pstree'*. *init* mengendalikan proses mana yang akan dijalankan dan dihentikan)

PID adalah nomor id dari proses

TTY adalah terminal/konsol (teletype) dari mana program tersebut dijalankan, semua proses yang dijalankan oleh *init* tidak mempunyai aturan terminal ini.

STAT status proses saat ini. keterangannya adalah

*S (sleeping)* → proses dalam keadaan sleep //kurang lebih 20 menit

*R (running)* → proses sedang berjalan

*D (defunct)* → proses tidak dipakai, merupakan proses yang berjalan sembarangan yaitu proses yang mati karena tidak ada yang berhubungan dengannya dan tidak ada fungsinya. Proses ini disebut juga sebagai proses “Zombie”, berjalan namun tidak ada fungsinya. Jika ditemukan proses seperti ini sebaiknya di kill secepat mungkin.

*I (idle)* → proses sedang tidak dijalankan/tidak dipakai

*Z (zombie)* → proses yang sudah mati, akan hilang saat di shutdown berikutnya, tidak berpengaruh pada sistem

*T (terminate)* → proses dihentikan

TIME menunjukkan cpu-time yang dihabiskan (akumulasi waktu yang digunakan proses), bukan waktu proses dijalankan

%CPU persentasi dari waktu CPU yang sudah digunakan, untuk prosesor modern proses akan berisi nol, kecuali untuk keperluan X-Window

%MEM persentasi memori sistem yang digunakan oleh proses, nilai berhubungan dengan RSS, bukan VSZ

VSZ ukuran memori virtual proses. besar memori proses termasuk shared libraries

RSS (resident set size) ukuran sebenarnya. Biasanya nilainya lebih kecil dari nilai yang diberikan 'top'. karena berbeda dalam hitungannya

CMD/COMMAND adalah nama program / perintah yang digunakan

**top**

Sintaks dari perintah top adalah:

```
top -hv | -bcHisS -d delay -n iterations
-p pid [, pid...]
```

Perintah **top** akan memberikan tampilan secara dinamis mengenai status proses yang sedang berlangsung/berjalan. Tampilan tabel proses yang ada akan di *update* pada interval waktu tertentu (default 5 detik). *option* yang penting pada **'top'** adalah **-d [detik]** untuk menentukan interval update dalam detik. Contoh:

```
top -d 2
```

```
top - 04:20:18 up 14 days, 12:38,  4 users,  load average: 0.16, 0.18, 0.15
Tasks:  89 total,   2 running,  87 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
Mem:    515756k total,   494992k used,    20764k free,    16940k buffers
Swap:  1951856k total,      128k used,  1951728k free,   166464k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      18   0  2952 1912  592 S    0  0.4   0:01.39 init
    2 root      10  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:01.10 migration/0
    4 root      34  19     0    0    0 S    0  0.0   0:01.20 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT  -5     0    0    0 S    0  0.0   0:00.64 migration/1
    7 root      34  19     0    0    0 S    0  0.0   0:00.07 ksoftirqd/1
    8 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/1
    9 root      10  -5     0    0    0 S    0  0.0   0:00.14 events/0
   10 root      10  -5     0    0    0 S    0  0.0   0:00.01 events/1
   11 root      10  -5     0    0    0 S    0  0.0   0:00.01 khelper
   31 root      10  -5     0    0    0 S    0  0.0   0:02.47 kblockd/0
   32 root      10  -5     0    0    0 S    0  0.0   0:00.05 kblockd/1
   33 root      13  -5     0    0    0 S    0  0.0   0:00.00 kacpid
   34 root      13  -5     0    0    0 S    0  0.0   0:00.00 kacpi_notify
  130 root      10  -5     0    0    0 S    0  0.0   0:00.00 kseriod
  155 root      15   0     0    0    0 S    0  0.0   0:00.11 pdflush
  156 root      15   0     0    0    0 S    0  0.0   0:01.60 pdflush
  157 root      10  -5     0    0    0 S    0  0.0   0:07.53 kswapd0
  209 root      11  -5     0    0    0 S    0  0.0   0:00.00 aio/0
  210 root      11  -5     0    0    0 S    0  0.0   0:00.00 aio/1
 1962 root      10  -5     0    0    0 S    0  0.0   0:00.00 ksuspend_usbd
 1964 root      10  -5     0    0    0 S    0  0.0   0:00.00 khubd
 2019 root      10  -5     0    0    0 S    0  0.0   0:00.00 ata/0
 2024 root      10  -5     0    0    0 S    0  0.0   0:00.00 ata/1
 2027 root      18  -5     0    0    0 S    0  0.0   0:00.00 ata_aux
 2091 root      13  -5     0    0    0 S    0  0.0   0:00.00 scsi_eh_0
 2092 root      10  -5     0    0    0 S    0  0.0   0:00.01 scsi_eh_1
 2292 root      10  -5     0    0    0 S    0  0.0   1:40.50 kjournald
 2452 root      15  -4  2436  684  412 S    0  0.1   0:00.27 udevd
 2799 www-data  15   0 26464  10m 4952 S    0  2.1   0:04.90 apache2
 3422 root      17  -5     0    0    0 S    0  0.0   0:00.00 kpsmoused
 3630 root      10  -5     0    0    0 S    0  0.0   0:01.72 kjournald
 3631 root      10  -5     0    0    0 S    0  0.0   0:02.35 kjournald
 4109 root      15   0  1692  484  416 S    0  0.1   0:00.01 getty
 4110 root      18   0  1696  488  416 S    0  0.1   0:00.01 getty
 4114 root      18   0  1692  484  416 S    0  0.1   0:00.00 getty
 4117 root      18   0  1696  484  416 S    0  0.1   0:00.00 getty
 4161 syslog    15   0  2012  840  652 S    0  0.2   0:34.60 syslogd
 4186 root      25   0  1836  516  432 S    0  0.1   0:00.00 dd
```

Ada beberapa field/kolom baru diantaranya

PRI Prioritas dari proses. Waktu penghitungan maksimum dalam milidetik untuk proses ini

NI Nilai 'nice', nilai prioritas yang diberikan secara manual.

**nice**

Semua proses memiliki hak yang sama dalam pembagian sumber daya, namun dapat diubah dengan perintah **nice** untuk memulai proses dengan prioritas yang diberikan. Sintaknya:

```
nice -n [value] [process]
```

keterangan:

Nilai negatif menambah prioritas hanya bisa dilakukan oleh 'root' namun nilai positif dapat dilakukan oleh siapa saja. Untuk mengubah prioritasnya dapat dilakukan lagi dengan perintah **renice.** Sintaknya:

```
renice [prioritas] [pid]
```

**kill**

Kadang ada proses yang tidak diperlukan kehadirannya dalam sistem atau terlalu banyak memakan resource komputer, sehingga akan menurunkan kinerjanya. Cara untuk menghentikan proses tersebut adalah dengan menggunakan perintah kill. Sintaknya:

```
kill -[signal] PID
```

menghentikan proses berdasarkan nomor ID proses, signal adalah nomor signal yang dapat digunakan, selain nomor signal dapat juga dengan menggunakan nama signalnya, misal untuk nomor signal **9** dapat diganti dengan **KILL**

```
killall -[signal] nama_proses
```

menghentikan proses berdasarkan nama prosesnya. Contoh:

**kill –9 253** → menghentikan proses dengan PID 253

**kill -KILL 253** → mematikan proses dengan PID 253

**killall httpd** → menghentikan proses dengan nama httpd

keterangan

Tanpa menyebutkan signal secara default akan diberikan signal **15 (sigterm)** yang akan menutup program "menunjukkan jalan keluar bagi program". **9 (sigkill)** akan mematikan program / mengeluarkan program dari sistem secara paksa (membasmi program) beberapa signal yang digunakan adalah:

- 1 HUP Hangup: menggantungkan proses
- 2 INT Interupt: mereboot program / proses
- 3 QUIT Quit: menutup program
- 6 ABRT abort: membatalkan proses
- 9 KILL Kill : mematikan proses (dengan paksa)
- 14 ALRM Alarm clock
- 15 TERM terminate: mengakhiri program

### 8.4.3. Hard Links dan Symbolic Links

Pada sistem operasi Linux ada istilah link dalam sistem file. Link berarti pointer atau penunjuk yang menunjuk ke file atau inode. Link ini berguna agar satu file dapat diacu dalam beberapa direktori lain yang berbeda.

Dengan adanya link ini memungkinkan data yang sama dapat dipakai oleh sejumlah pengguna dalam suatu jaringan multiuser, dan sebuah file dapat dianggap sama dengan file lain tanpa melakukan penyalinan, yang tentunya akan memakan banyak ruang media penyimpanan.

Terminologi link dalam Linux mempunyai kesamaan dengan shortcut pada Windows. Hanya saja, pengguna Linux diperbolehkan melakukan *cd* pada link tersebut jika link itu adalah sebuah

direktori, sementara pada Windows tidak dapat.
Untuk membuat link, Unix/Linux menyediakan utilitas bernama *ln* dengan sintaks:

**$ ln [option] file_sumber file_target**

Ada dua macam kategori link, yaitu hard link dan symbolic link.

### Hard link

Yaitu dua file yang menuju pada inode yang sama. Misalnya:
Pertama buat file teks dengan nama *data.txt* dengan perintah *cat*. Ketikkan isi file lalu tekan *Ctrl+D*.

```
$ cat > data.txt
ini isi file data
$
```

Buat link pada file *data.txt* dengan perintah *ln*.

```
$ ls -l
-rw-r--r-- 1 user user 18 2007-12-19 04:01 data.txt
drwxr-xr-x 2 user user 4096 2007-12-17 21:35 Desktop
$ ln data.txt percobaan
$ cat percobaan
ini isi file data
$ ls -l
total 12
-rw-r--r-- 2 user user 18 2007-12-19 04:01 data.txt
drwxr-xr-x 2 user user 4096 2007-12-17 21:35 Desktop
-rw-r--r-- 2 user user 18 2007-12-19 04:01 percobaan
$
```

Setelah dilakukan *ln* file *"data.txt"* di link ke file *"percobaan"*, maka file percobaan akan berisi sama persis dengan file *"data.txt"*. Disitu juga terlihat jumlah link ke inode ada dua buah. Kita bisa mencoba dengan perintah:

```
$ ls -i
187412 data.txt 159367 Desktop
187412 percobaan
$
```

Terlihat bahwa file *"data.txt"* dan file *"percobaan"* mengarah pada inode yang sama. Perintah *ln* juga mengizinkan kita untuk mengaitkan dua buah file yang terletak pada partisi yang berbeda.

Penghapusan terhadap file satu tidak akan menyebabkan file dua terhapus, demikian pula sebaliknya.

### Symbolic Link

Symbolic link dapat dibuat dengan menyertakan opsi *-s* pada perintah *ln*. Pada symbolic link, file target hanya merupakan pointer yang mengarah ke file sumber. Kalau file sumber dihapus, maka file target juga tidak akan berfungsi lagi karena pada symbolic link, yang mengarah pada inode hanya file sumber.

Untuk lebih jelasnya, perhatikan contoh di bawah ini:

```
$ ls -l
total 12
-rw-r--r-- 2 user user 18 2007-12-19 04:01 data.txt
drwxr-xr-x 2 user user 4096 2007-12-17 21:35 Desktop
```

```
-rw-r--r-- 2 user user 18 2007-12-19 04:01 percobaan
$ ln -s data.txt percobaan2
$ ls -l
total 12
-rw-r--r-- 2 user user 18 2007-12-19 04:01 data.txt
drwxr-xr-x 2 user user 4096 2007-12-17 21:35 Desktop
-rw-r--r-- 2 user user 18 2007-12-19 04:01 percobaan
lrwxrwxrwx 1 user user 8 2007-12-19 04:06 percobaan2 -> data.txt
$ ls -i
187412 data.txt 159367 Desktop 187412 percobaan 187414 percobaan2
$ cat data.txt
ini isi file data
$ cat percobaan2
ini isi file data
$
```

Jika file *data.txt* dihapus, file percobaan tidak ikut hilang, tapi akan tidak berfungsi lagi karena dia mengarah pada file yang tidak ada.

### *8.4.4.* Manajemen pengguna *(user)*

Admin harus membuat account pengguna yang biasa untuk pemakaian sehari-hari, dan menggunakan *root* hanya untuk pekerjaan administrasi sistem. Untuk membuat sebuah pengguna, dapat menggunakan aplikasi yang disertakan pada distribusi Linux, atau dengan mengedit file password secara manual.

#### Script yang Tersedia

Cara termudah untuk mengelola pengguna dan grup adalah dengan script dan program yang tersedia. Linux menyertakan program `adduser`, `userdel`, `chfn`, `chsh`, dan `passwd` untuk berhadapan dengan pengguna. Perintah `groupadd`, `groupdel`, dan `groupmod` digunakan untuk berhadapan dengan grup. Selain `chfn`, `chsh`, dan `passwd`, program-program tadi hanya dijalankan sebagai `root`, dan terletak pada `/usr/sbin`. `chfn`, `chsh`, dan `passwd` dapat dijalankan oleh siapapun dan terletak pada `/usr/bin`.

Pengguna bisa ditambahkan dengan program *adduser*. Kita akan mulai dengan keseluruhan prosedur, menampilkan seluruh pertanyaan yang ditanyakan dan deskripsi singkat tentang artinya. Jawaban default adalah yang terletak dalam kurung siku, dan dapat dipilih untuk hampir semua pertanyaan, kecuali hendak mengubah sesuatu.

```
# adduser
Login name for new user []:
tekaje
```

Ini adalah nama pengguna yang akan digunakan untuk login. Biasanya, nama login terdiri dari delapan karakter atau kurang, dan semua adalah huruf kecil. (dapat menggunakan lebih dari delapan karakter, atau menggunakan angka, tetapi hindari itu kecuali dengan alasan yang cukup penting)

Nama login juga bisa digunakan sebagai argumen pada perintah baris:

```
# adduser tekaje
```

Pada semua kasus, setelah menyediakan nama login, *adduser* akan meminta ID pengguna::

```
User ID ('UID') [ defaults to
next available ]:
```

ID Pengguna (UID) adalah bagaimana kepemilikan ditentukan pada Linux. Setiap pengguna memiliki angka unik, dimulai dari 1000 pada beberapa distribusi linux. Sebuah UID juga dapat dipilih untuk pengguna baru, atau dengan membiarkan *adduser* memberikan nilai berikutnya yang masih kosong.

```
Initial group [users]:
```

Semua pengguna diletakkan pada grup *users* secara default. Pengguna baru dapat diletakkan pada grup yang berbeda, tetapi tidak disarankan kecuali admin mengetahui apa yang dilakukannya.

```
Additional groups (comma separated) []:
```

Pertanyaan ini mengijinkan untuk meletakkan pengguna baru pada grup tambahan. Dimungkinkan sebuah pengguna terdapat pada beberapa grup pada saat yang bersamaan. Hal ini berguna jika anda memiliki grup untuk beberapa hal seperti memodifikasi file-file web site, menjalankan permainan, dan selanjutnya. Sebagai contoh, beberapa situs mendefinisikan grup *wheel* sebagai satu-satunya grup yang dapat menggunakan perintah *su*.

```
Home directory [/home/tekaje]
```

Direktori home default diletakkan pada */home*. Sangat memungkinkan untuk mengubah direktori *home* pada lokasi yang berbeda (atau pada banyak lokasi). Langkah ini mengijinkan untuk menentukan lokasi direktori *home*.

```
Shell [ /bin/bash ]
```

bash adalah shell default untuk sebagian besar distribusi Linux, dan cukup baik untuk sebagian besar orang. Pada dialog ini menentukan shell untuk pengguna. Pada masa yang akan datang pengguna dapat mengubahnya dengan perintah *chsh*

```
Expiry date (YYYY-MM-DD) []:
```

Account dapat ditentukan untuk kadaluarsa pada tanggal yang ditentukan. Secara default, tidak ada tanggal kadaluarsa. Opsi ini cukup berguna untuk orang-orang yang menjalankan sebuah ISP yang ingin agar sebuah account kadaluarsa pada tanggal tertentu, kecuali mereka menerima pembayaran tahun berikutnya.

```
New account will be created as follows:
---------------------------------------
Login name:  tekaje
UID:  [ Next available ]
Initial group: users
Additional groups: [ None ]
Home directory: /home/tekaje
Shell:  /bin/bash
Expiry date: [ Never ]
```

Jika ingin keluar, tekan **Control**+**C**. Selain itu, tekan ENTER untuk melanjutkan dan membuat account.

Sekarang terlihat semua informasi yang telah dimasukkan tentang account baru dan diberikan kesempatan untuk

membatalkan pembuatan account baru. Jika terdapat informasi yang salah, tekan **Control**+**C** dan ulangi langkahnya. Selain itu, tekan enter dan account akan dibuat.

```
Creating new account...

Changing the user information for tekaje
Enter the new value, or press return for the default
 Full Name []: Siswa
 Room Number []: Sekolah 130
 Work Phone []:
 Home Phone []:
 Other []:
```

Semua informasi ini opsional dan pengguna bisa menggantinya menggunakan *chfn*.

```
Changing password for tekaje
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and
numbers.
New password:
Re-enter new password:
Password changed.

Account setup complete.
```

Berikutnya adalah memasukkan sebuah kata sandi untuk pengguna baru. Biasanya, jika pengguna baru tidak hadir pada saat ini, cukup diberikan kata sandi default dan memberitahukan pengguna untuk menggantinya dengan sesuatu yang lebih aman.

*Memilih sebuah Katasandi*: Memiliki kata sandi yang aman adalah baris pertahanan pertama untuk mengindari peristiwa *cracking*. Semua orang tidak ingin memiliki kata sandi yang mudah ditebak, karena hal itu akan mempermudah seseorang untuk memasuki sistem yang dimilikinya. Idealnya, sebuah kata sandi yang aman berupa kata acak, termasuk huruf besar dan kecil, angka, dan karakter acak. (Karakter tab mungkin bukan pilihan yang bijak, tergantung dari komputer digunakan untuk login.) Terdapat banyak paket perangkat lunak yang dapat menghasilkan kata sandi acak; cari Internet untuk utilitas ini.

Secara umum, gunakan naluri: jangan pilih kata sandi yang berhubungan dengan ulang tahun seseorang, frase umum, sesuatu yang dapat ditemukan pada meja kerja kita, atau sesuatu yang berhubungan dengan pengguna.

Menghapus pengguna tidaklah sulit sama sekali. Cukup jalankan `userdel` dengan nama account yang hendak dihapus. Account tersebut harus diverifikasi sedang tidak login, dan tidak ada proses yang berjalan atas pengguna tersebut. Juga, ingat bahwa setelah menghapus pengguna tersebut, semua informasi pengguna tersebut juga hilang secara permanen.

```
# userdel tekaje
```

Perintah ini menghapus pengguna `tekaje` dari sistem. Pengguna dihapus dari `/etc/passwd`, `/etc/shadow`, dan `/etc/group`, tetapi tidak menghapus direktori *home* pengguna.

Jika hendak menghapus direktori homenya, gunakanlah perintah ini:

```
# userdel -r tekaje
```

Program untuk menambahkan dan menghapus grup sangatlah sederhana. `groupadd` akan menambahkan satu daftar pada file `/etc/group` dengan ID grup yang unik, sementara `groupdel` akan menghapus grup yang ditentukan. Untuk menambahkan sebuah grup bernama *cvs*:

```
# groupadd cvs
```

dan untuk menghapusnya:

```
# groupdel cvs
```

### 8.4.5. Mengganti Kata Sandi

Program `passwd` mengganti kata sandi dengan memodifikasi file `/etc/shadow`. File ini menyimpan semua kata sandi untuk sistem dalam format yang terenkripsi. Untuk mengubah kata sandi, ketikkan:

```
$ passwd
Changing password for chris
Old password:
Enter the new password (minumum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and
numbers.
New password:
```

Seperti yang terlihat, kata sandi lama akan diminta untuk dimasukkan. Kata sandi tidak akan terlihat ketika diketikkan, sama seperti ketika login. Berikutnya masukkan kata sandi baru. `passwd` melakukan banyak pengujian pada kata sandi baru, dan akan keberatan jika kata sandi baru tersebut tidak lolos dalam pengujiannya. Peringatan ini bisa saja diabaikan jika perlu. Setelah itu akan diminta untuk memasukkan kata sandi kedua kalinya untuk konfirmasi.

Dengan akses sebagai `root`, pengguna bisa mengganti kata sandi pengguna lain:

```
# passwd tedi
```

Prosedur yang sama seperti diatas akan dilalui, kecuali memasukkan kata sandi lama pengguna. (Satu dari banyak keuntungan sebagai `root`...)

Jika diperlukan, `root` bisa menonaktifkan sebuah account untuk sementara, dan mengaktifkannya kembali jika diperlukan. Mengaktifkan dan menonaktifkan sebuah account dapat dilakukan dengan `passwd`. Untuk menonaktifkan sebuah account, lakukan perintah ini sebagai `root`:

```
# passwd -l asep
```

Hal ini akan mengganti kata sandi *asep* dengan nilai enkripsi yang tidak akan sama. Account dapat diaktifkan dengan:

```
# passwd -u asep
```

Sekarang, account *asep* kembali normal. Menonaktifkan sebuah account dapat berguna jika pengguna tidak mematuhi aturan yang telah dibuat pada sistem, atau jika mereka mengekspor banyak salinan dari `xeyes(1)` pada desktop X.

### 8.4.6. Manajemen Kepemilikan

Sistem file menyimpan informasi kepemilikan dari setiap file dan direktori pada sistem. Hal ini juga termasuk pengguna dan grup yang memiliki file tertentu. Cara termudah untuk melihat informasi ini adalah dengan perintah `ls`:

```
$ ls -l /usr/bin/wc
-rwxr-xr-x 1 root bin 7368 Jul 30
1999 /usr/bin/wc
```

Perhatikan kolom ketiga dan keempat. Kolom ini berisi nama pengguna dan nama grup yang memiliki file ini. Terlihat bahwa pengguna *“root”* dan grup *“bin”* memiliki file ini.

Kepemilikan sebuah file dapat diganti dengan perintah `chown` (yang berarti “change owner (ganti pemilik)”) dan `chgrp` (yang berarti “change group (ganti grup)”). Untuk mengganti pemilik file menjadi *daemon*, kita menggunakan `chown`:

```
# chown daemon /usr/bin/wc
```

Untuk mengganti pemilik grup kepada *“root”*, gunakan `chgrp`:

```
# chgrp root /usr/bin/wc
```

Gunakan juga `chown` untuk menentukan nama pengguna dan grup sebagai pemilik sebuah file:

```
# chown daemon:root /usr/bin/wc
```

Pada contoh diatas, pengguna dapat menggunakan tanda titik dan bukannya titik dua. Hasilnya adalah sama; namun titik dua dianggap bentuk yang lebih baik. Penggunaan titik sudah ditinggalkan dan mungkin akan dihapus pada versi baru dari `chown` untuk mengijinkan penggunaan titik pada nama pengguna. Nama pengguna ini cenderung populer dengan Windows Exchange Servers dan dijumpai pada alamat email pada umumnya seperti: *mr.jones@example.com*. Pada Linux, administrator disarankan untuk menjauhi nama pengguna seperti itu karena beberapa script masih menggunakan titik untuk mengindikasikan sebuah pengguna dan grup dari sebuah file atau direktori. Pada contoh, `chmod` akan menginterpretasikan *mr.jones* sebagai pengguna *“mr”* dan grup *“jones”*.

Kepemilikan file sangatlah penting dalam penggunaan sistem Linux, meskipun statusnya hanyalah seorang pengguna. Pengguna seringkali harus memperbaiki kepemilikan pada file atau node device.

### 8.4.7. Kontrol Akses terhadap File

Setiap file dan direktori pada sistem operasi Linux memiliki pemilik dan hak akses yang berbeda-beda. Pemilik file bisa mengubah hak akses yang melekat pada file dan direktori tersebut.

#### Mengubah Kepemilikan File

Suatu file atau direktori dapat diubah pemiliknya dengan menggunakan

perintah chown, asal kita memiliki hak untuk mengubahnya.

Misalnya perintah berikut akan mengubah file *data.txt* menjadi milik user *joko*.

```
# chown joko data.txt
```

**Mengubah Group File**

Suatu file atau direktori dapat diubah grupnya dengan menggunakan perintah chgrp, asal kita memiliki hak untuk mengubahnya.
Misalnya perintah berikut akan mengubah file *data.txt* menjadi dapat diakses oleh user-user yang berada pada grup support.

```
# chgrp support data.txt
```

**Mengubah Izin Hak Akses**

Untuk mengubah izin akses file atau direktori, gunakan perintah chmod. Ada dua cara untuk mengubah izin akses file, yaitu dengan notasi angka dan notasi huruf.
Sintaks perintahnya adalah sebagai berikut:

chmod <whoXperm> <file>

dengan:

- *who* adalah u (user atau pemilik), g (grup), dan o (other)
- X adalah tanda + atau -
- *perm* adalah r (read), w (write), x (execute)

Contoh:

```
# chmod u+x data.txt
```

Perintah di atas berfungsi untuk mengeset file *data.txt* menjadi *executable* terhadap pemiliknya.

```
# chmod go-wx data.txt
```

Perintah di atas berfungsi untuk menghilangkan hak *write* dan *execute* dari grup dan other.

```
# chmod ugo+rwx data.txt
```

Perintah di atas berfungsi untuk memberikan hak *read, write,* dan *execute* kepada semua orang.
Selain dengan cara-cara di atas kita bisa juga mengubah izin akses file dengan notasi angka.

Misalnya:

```
# chmod 755 data.txt
```

Perintah di atas berfungsi untuk mengubah izin akses file menjadi *rwxr-xr-x*. Angka 755 dapat diterjemahkan menjadi bit seperti berikut ini:

- rwx = 111, yang berarti nilai 7 desimal
- r-x = 101, yang berarti nilai 5 desimal
- r-x = 101, yang berarti nilai 5 desimal

### 8.4.8. Otomasi dan Penjadwalan Pekerjaan

Jika anda harus melakukan pekerjaan-pekerjaan secara rutin, mengapa anda masih harus melakukannya dengan mengetikkan baris-baris perintah yang panjang dan rumit. Di Linux anda dapat mengatur pekerjaan-pekerjaan tersebut secara otomatis/penjadwalan otomatis dengan menggunakan cron. Dengan demikian anda dapat menghemat waktu anda, karena pekerjaan-pekerjaan rutin

yang harus anda lakukan, diambil alih oleh sistem komputer anda, bahkan setelah itu anda dapat melupakannya.

Di lingkungan UNIX ataupun Linux penjadwalan kerja otomatis yang dilakukan oleh sistem bukan merupakan barang baru, karena kita tahu bahwa Linux ataupun UNIX merupakan sistem operasi yang *multitasking* dan multiuser, sehingga sangat tepat digunakan untuk menjalankan berbagai operasi yang bersifat *critical mission.*

Secara singkat *cron* dijalankan dengan menggunakan perintah *crontab*. *Crontab* sendiri akan menyimpan baris-baris perintah tersebut pada direktori */var/spool/cron/crontab*. Untuk dapat menjalankan *crontab*, pastikan bahwa sistem Linux anda telah menjalankan daemon yang bernama *crond* pada waktu booting. Pada dasarnya tidak ada perintah yang bernama *cron*, akan tetapi anda hanya menggunakan utilitas *crontab* dan daemon *crond*.

Sebagai contoh sebelum penulis mengenal *cron*, penulis harus mengetikkan baris-baris perintah setiap harinya untuk menghapus file-file temporer yang berada pada direktori */tmp*. Memang pada beberapa distribusi Linux akan menghapus file-file temporer setiap kali komputer anda melakukan booting. Akan tetapi jika komputer anda harus hidup selama berhari-hari, maka file-file temporer anda akan semakin banyak menumpuk.

Misalkan anda harus menghapus file-file yang sudah berusia lebih dari tiga hari pada direktori /tmp, anda dapat mencarinya dengan menggunakan perintah *find*, dengan target direktori adalah /tmp, dengan menambahkan beberapa option agar perintah *find* langsung menemukan file-file yang berusia lebih dari tiga hari. Option *-exec* digunakan untuk mengeksekusi setiap perintah yang diberikan jika file-file yang dimaksud telah ditemukan.

```
# find /tmp \! -type d -atime +3
-exec rm {} \;
```

Perintah ini akan meminta *find* mengeksekusi perintah rm, string {} digunakan untuk menampilkan keseluruhan hasil dari baris-baris perintah sesuai dengan option yang diberikan.

Dengan demikian, sekarang anda sudah dapat menghapus file-file temporer pada direktori */tmp*, tinggal sekarang anda harus menentukan kapan baris-baris perintah tersebut akan dijalankan oleh cron, anda harus melakukannya dengan crontab yang memiliki format yang terdiri dari 6 buah field, yaitu:

```
minute hour day month dayofweek
command
```

Berikut adalah penjelasan dari field-field tersebut:

1. Minute = Menit (bernilai 0 sampai dengan 59).
2. Hour = Jam (bernilai antara 0 sampai 23).
3. Day = Hari (bernilai antara 1 sampai 31).
4. Month = Bulan (1 sampai 12 atau berupa nama bulan jan, feb, mar dsb).
5. dayofweek = data hari mingguan (0 sampai 6, 0=Minggu)
6. Command = Perintah

Berikut ini adalah sebuah contoh entri pada *cron* dengan *command* berupa shell script yang berjalan dengan Bourne Shell. Entri ini akan dapat dijalankan jika field 1 sampai dengan 5 bernilai benar

```
0 4 23 feb, jun sun sh runcron
```

Dua field pertama menunjukkan waktu pukul 04.00. Field ketiga berarti tanggal 23, untuk field 4 dan 5 menunjukkan bulan Februari dan Juni. Contoh ini memang tidak realistis, karena perintah akan dijalankan hanya pada setiap hari minggu yang jatuh pada tanggal 23 bulan Februari dan Juni.

Agar dapat menjalankan perintah pada entri `cron` agar dijalankan setiap hari anda dapat memberikan tanda asterik (*) pada field 3, 4 dan 5 sehingga perintah yang anda berikan tidak dibatasi oleh kondisi field 3,4 dan 5. Berikut adalah contoh pada entri tersebut

```
0 4 * * * find /tmp -atime 3 -
exec rm {} \;
```

Jika anda menginginkan perintah tersebut dijalankan setiap diawal bulan, tambahkan angka 1 pada field ketiga sehingga menjadi:

```
0 4 1 * * find /tmp -atime 3 -
exec rm {} \;
```

Berikut ini adalah contoh entri yang akan dijalankan setiap tanggal 1 dan 20 tiap bulan:

```
0 4 1,20 * * find /tmp -atime 3
-exec rm {} \;
```

Sedangkan untuk menjalankan `cron` agar dieksekusi setiap tanggal 1 sampai dengan tanggal 15 tiap bulannya, seperti pada entri berikut ini:

```
0 4 1-15 * * find /tmp -atime 3
-exec rm {} \;
```

Dengan memberikan tanda slash (/) pada entri ketiga akan membuat perintah dijalankan setiap 3 hari seperti contoh berikut ini:

```
0 4 */3 * * find /tmp -atime 3 -
exec rm {} \;
```

Setelah itu yang harus anda lakukan adalah meletakkan entri tersebut didalam file *crontab*. Pada prompt lakukan perintah berikut ini:

```
# crontab – e
```

Perintah diatas langsung akan membuka editor *vi*, bagi anda yang menggunakan *emacs*, sebelum anda mengetikkan perintah `crontab -e`, pada prompt anda ketikkan perintah berikut (bourne shell):

```
# export VISUAL = emacs.
```

Sedangkan pada `c` shell ketikkan perintah berikut ini:

```
# setenv VISUAL emacs.
```

Setelah editor terbuka, tuliskan entri untuk `cron` seperti pada contoh berikut ini:

```
# Menghapus file-file temporer
yang berusia lebih dari tiga
hari

# pada direktori /tmp

0 4 * * * find /tmp -atime 3 -
exec rm {} \;
```

Setelah keluar dari *vi*, secara otomatis akan tersimpan menjadi file *crontab*. Untuk melihat entri *crontab* anda ketikkan perintah *crontab – l* pada prompt anda.

## 8.5. Penggunaan dan Pengaturan Halaman Manual

### 8.5.1. Halaman manual

Perintah `man` (kependekan dari "manual") adalah bentuk tradisional dari dokumentasi online pada sistem operasi Unix dan Linux. Terdiri dari file-file yang diformat secara khusus, "halaman man", ditulis dari program-program besar dan didistribusikan didalam perangkat lunak itu sendiri. Menjalankan `man` *perintahTertentu* akan menampilkan halaman man dari (secara alami) perintah tertentu, seperti pada contoh kami adalah program imajinatif *perintahTertentu*.

Seperti yang bisa anda bayangkan, jumlah dari halaman manual terus bertambah dengan cepat, menjadi membingungkan dan rumit, bahkan untuk pengguna ahli. Sehingga, untuk alasan inilah, halaman manual dikelompokkan pada beberapa bagian. Sistem ini telah ada untuk waktu yang cukup lama; cukup lama sehingga anda akan sering melihat perintah, program, bahkan fungsi pustaka program merujuk pada nomor bab dari manual.

Sebagai contoh anda mungkin melihat sebuah referensi ke `man(1)`. Penomoran ini memberitahu anda bahwa "man" didokumentasikan pada bagian 1 (Perintah pengguna); Anda bisa menentukan bahwa anda ingin manual bagian 1 dari perintah "man" dengan perintah:

```
man 1 man
```

Menentukan bagian yang harus dicari oleh man berguna pada kasus beberapa hal dengan nama yang sama.

### 8.5.2. Whatis

Perintah `whatis` memberikan deskripsi jelas tentang perintah sistem, dengan gaya referensi perintah saku.

Contoh:

```
$ whatis whatis
whatis (1) - search the whatis
database for complete words
```

### 8.5.3. Apropos

Perintah `apropos` digunakan untuk mencari halaman manual yang berisi kata kunci yang diberikan.

Contoh:

```
$ apropos wav
cdda2wav (1) - a sampling utility that dumps CD audio data into wav
sound files
netwave_cs (4) - Xircom Creditcard Netwave device driver
oggdec (1) - simple decoder, Ogg Vorbis file to PCM audio file (WAV or
RAW)
wavelan (4) - AT&T GIS WaveLAN ISA device driver
wavelan_cs (4) - AT&T GIS WaveLAN PCMCIA device driver
wvlan_cs (4) - Lucent WaveLAN/IEEE 802.11 device driver
```

## 8.6. Backup Data

### 8.6.1. Alasan Melakukan Backup

Sebelum melaksanakan proses backup data, perlu diketahui beberapa alasan mengapa backup data perlu dilakukan. Ada pun alasan melakukan proses backup data diantaranya adalah sebagai berikut:

- Kegagalan pada perangkat penyimpanan seperti harddisk
- Bug pada perangkat lunak dapat mengakibatkan kerusakan file (file corrupt)
- Kesalahan konfigurasi oleh administrator
- Ketidak sengajaan menghapus atau menulis ulang file (penggunaan `rm`, `mv` atau `cp`)
- Penghapusan oleh malicious software atau perangkat lunak
- Pencurian mesin dengan harddisk berisi data di dalamnya
- Kebakaran atau bencana lainnya yang dapat mengakibatkan kerusakan pada data.

### 8.6.2. Media Backup

Terdapat beberapa media backup data yang dapat digunakan, sebagai berikut:

- Pita magnetik atau tape
- Harddisk (harddisk-to-harddisk copy)

  Menggunakan metode ini dapat memiliki resiko kehilangan baik hasil backup atau pun aslinya namun metode ini baik digunakan untuk *remote computer.*
- CD atau DVD

  Penggunaan media ini baik untuk penyimpanan jangka waktu yang lama dan mudah untuk dibawa ke lokasi komputer remote.

## 8.7. Soal-soal latihan

1. Apakah yang dimaksud dengan shell?
2. Apakah perbedaan antara C shell, korn shell, dengan bash?
3. Wildcard apa yang digunakan untuk mencari file yang berekstensi.iso?
4. File apa yang digunakan oleh bash untuk mendefinisikan variabel-variabel?
5. File apa yang digunakan bash untuk mencatat program atau perintah yang telah dituliskan?
6. Apakah fungsi dari alias?
7. Perintah apa yang digunakan untuk memperlihatkan isi suatu direktori lengkap dengan atributnya?
8. Bagaimana cara membuat empat subdirektori baru dalam satu baris perintah?
9. Bagaimana cara menggabungkan sejumlah file ke dalam satu file baru?
10. Perintah apa yang digunakan untuk memasukkan suatu baris kata pada akhir baris suatu file?
11. Perintah apa yang digunakan untuk menyalin suatu direktori secara rekursif(tuliskan sintaksnya!)?
12. Apakah fungsi dari touch?
13. Perintah apa yang digunakan untuk menghapus direktori dan subdirektori yang berada di dalamnya?
14. Bagaimanakah cara membuat hardlink?
15. Apa perbedaan antara hardlink dan symbolic link?
16. Apa perbedaan tanda > dengan >> pada proses redirection?
17. Apakah fungsi dari tanda “|” ?
18. Apa fungsi dari perintah ps?
19. Dimanakah letak file untuk penjadwalan disimpan?
20. Apa aplikasi yang digunakan untuk menjadwalkan proses?
21. Perintah apa yang digunakan untuk mematikan proses yang sedang berjalan?

22. Bagaimana cara mengubah sandi/password user?
23. Bagaimana cara mengubah pemilik dari suatu file atau direktori?
24. apakah arti atribut rwxr--r--?
25. Perintah apa yang digunakan untuk membuat suatu file menjadi executable?
26. Apakah fungsi dari perintah help?
27. Bagaimana cara memanggil manual dari suatu aplikasi?
28. Apakah fungsi dari perintah Apropos?
29. Mengapa Back Up data harus dilakukan?
30. Apa saja media backup yang dapat digunakan? Sebutkan Perbedaannya?