Backup dan Restore dengan Partimage

Pernahkah Anda kehilangan data? Atau saat Anda sangat membutuhkan data, komputer yang Anda gunakan rusak? Jika Anda telah memiliki sistem *backup*, dalam beberapa menit, Anda sudah bisa bekerja normal kembali. Dengan Linux, Anda dapat menggunakan *partimage* secara mudah untuk melakukan backup dan *restore* data Linux maupun Windows.

Sistem backup juga berguna bila kita ingin mencoba beberapa sistem operasi, misalnya Linux Debian, Fedora, Slackware, Win XP, Win ME, Win 98, dan lain-lain, sedangkan kapasitas komputer kita terbatas. Tutorial ini menjadi hal yang menarik, yang diharapkan bisa menambah semangat Anda untuk belajar Linux. Dalam tutorial ini, kita mencoba membuat backup dengan Partimage, dan juga melakukan restore dengan Partimage.

Apa itu Partimage?

Partimage (*partition image*) adalah utilitas Linux/UNIX yang akan menyimpan partisi dalam banyak format menjadi image file. Image file ini dapat dimampatkan (kompresi) ke GZIP/BZIP2, dan juga bisa dipecah menjadi beberapa file. Image juga bisa disimpan melalui jaringan (sejak versi 0.6.0). Lisensi Partimage adalah *free software* GPL 2 (GNU *General Public License*).

Partimage hanya meng-*copy* blok data yang digunakan oleh partisi. Untuk efisiensi dan kecepatan, blok yang tidak digunakan tidak disimpan dalam image file. Ini tidak seperti kalau menggunakan 'dd' command, yang akan menyimpan juga blok yang tidak digunakan.

Beberapa keuntungan menggunakan Partimage:

- Kita dapat me-restore partisi Linux, Windows atau sistem file lainnya, misalnya jika ada masalah virus, file *system error, manipulation error*, dan lain-lain.
- Image file yang dibuat bisa digunakan untuk banyak komputer yang identik. Misalnya dalam satu tempat terdapat 100 komputer, maka dengan membuat satu image file bisa dipakai untuk semua komputer kalau terjadi masalah.
- Bisa membuat berbagai image file dalam satu komputer, karena bentuknya berupa file.

Beberapa file system yang di-*support* oleh Partimage ada pada Tabel 1.

NTFS (Windows NT File System) tidak di-support penuh untuk beberapa hal berikut ini:

1. Jika system files terfragmentasi (*fragmented*). Untuk mengatasi hal ini, maka sebaiknya partisi NTFS di-*defrag* terlebih dahulu, setelah itu baru di-backup.
2. Jika system files di-*compres*. Untuk mengatasi hal ini, maka sebaiknya partisi NTFS dibuat tanpa kompresi (*decompression*) terlebih dahulu, setelah itu baru di-backup.

Beberapa distribusi Linux sudah menggunakan Partimage, misalnya Kanotix, Knoppix, dan SystemRescueCD. Jika distro Anda belum memilikinya, harap lihat dan download informasi terbarunya ke *http://www.partimage.org/Main_Page*.

Apa kompresi yang digunakan?

Ketika akan mem-backup ke image file, Anda dapat memilih empat level kompresi berikut ini:

1. *none* atau tanpa kompresi.
   Dengan *No compression* akan dihasilkan image file yang besar, sebesar ukuran data yang digunakan. Ini diperlukan kalau *space* kita besar. Pada pilihan ini, proses backup dan restore akan sangat cepat.
2. gzip atau image dikompresi dengan gzip (*default*).
   Dengan Gzip compression, didapatkan image file yang kecil. Pilihan ini banyak digunakan, sehingga dipasang secara

Tabel 1.

Nama	Deskripsi	Status
ext2fs/ext3fs	linux File System standar	stable
Reiser3	a journalized and powerful file system	stable
FAT16/32	DOS and Windows file systems	stable
HPFS	IBM OS/2 File System	stable
JFS	Journalised File System, dari IBM	stable
XFS	another jounalized and efficient File System, dari sgi	stable
UFS	Unix File System	beta
HFS	MacOS File System	beta
NTFS	Windows NT, 2000 and XP	experimental

default. Pilihan ini akan membutuhkan waktu lama, tetapi akan menghasilkan ukuran file yang kecil.

3. Bzip2 atau image dikompresi dengan bzip2.
   Dengan Bzip2 compression, akan didapatkan ukuran image file yang sangat kecil, namun ini akan menjadikan proses lebih lambat lagi. Ini akan berguna jika ruang penyimpanan backup kita sangat terbatas, dan kita punya banyak waktu.
4. Lzo atau image dikompresi dengan lzo (untuk versi 0.7.0 ke atas).
   Lzo compression merupakan kompresi yang lebih cepat dari pada gzip, tetapi efisiensinya agak rendah sedikit dari gzip.

Penulis telah mencoba di lingkungan Linux dan Windows (backup kanotix knoppix suse winxp win me dalam satu komputer), sehingga bila ingin mencoba kanotix, tinggal restore beberapa menit (sekitar 5 menit). Mau mencoba Suse, tinggal restore, dan seterusnya.

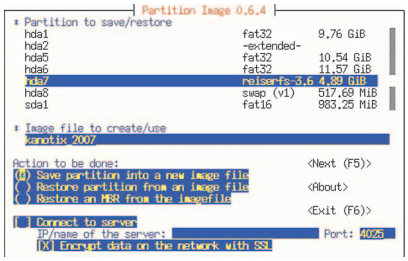
Demikian juga bila terjadi *hang* pada Windows, dengan mudah kita me-restore (5 sampai 10 menit). Bayangkan bila kita harus menginstal Windows, dan aplikasi lainnya dari awal, bisa-bisa satu hari kalau untuk semua aplikasi.

Tentunya kita harus memisahkan antara partisi OS, dengan partisi data atau *master* backup, sehingga lebih aman bila hanya menempatkan data pada partisi yang terpisah dengan OS (misalnya partisi pertama untuk Windows, partisi kedua untuk data, dan partisi lainnya untuk Linux).

Backup dengan Partimage

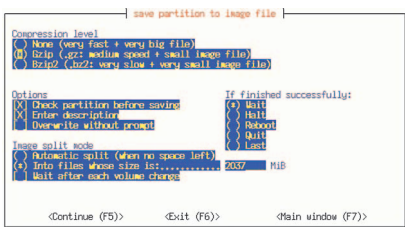
Beberapa langkah yang harus diambil untuk mem-backup adalah:

1. Optimalkan sistem.
   Sistem yang akan di-backup adalah yang sistem optimal (menurut kita sendiri). Artinya, di saat sistem bekerja normal, aplikasi yang dipasang juga sudah dipilih, dan di-setting dengan baik. Menata system sampai optimal sebelum di-backup adalah sangat penting.
2. *umount* partisi yang akan di-backup. Setelah sistem optimal, partisi sistem di-umount untuk di-backup. Kalau kita mem-backup sistem Linux, maka kita bisa memanfaatkan Linux live-cd, misalnya Kanotix, Knoppix, dan Rescuecd. Kalau kita mem-backup Windows, kita bisa melakukannya melalui Linux yang sudah terinstal di harddisk ataupun melalui Linux live CD. Berikut contoh perintah umount:
   ```
   # umount /dev/hda7
   ```
3. mount partisi penyimpanan, dan buka partisi.
   Siapkan partisi yang akan digunakan untuk menyimpan image file. misalnya:
   ```
   # mount /dev/hda5 /media/
   hda5
   ```
   Lalu buka partisi tersebut:
   ```
   # cd /media/hda5
   ```
4. Aktifkan Partimage dengan jalan mengetikkan perintah Partimage:
   ```
   # partimage
   ```
   Pada layar akan muncul dialog seperti Gambar 1.



Gambar 1. Backup.

5. Dari Gambar 1, tentukan letak partisi yang akan di-backup. Dalam contoh ini, dipilih hda7. Lalu tulis nama file image yang akan dibuat (misal: kanotix_2007). Karena mem-backup, maka dipilih *save partition* . Lakukan seperti di atas, kemudian untuk proses berikutnya tekan F5, maka akan muncul seperti Gambar 2.

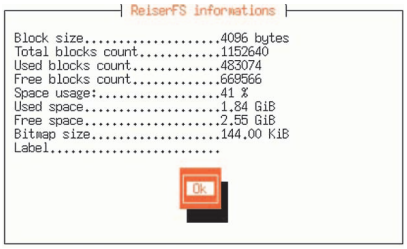


Gambar 2. Option Backup.

6. Dari Gambar 2, Anda bisa memilih beberapa *option* untuk kompresi, deskripsi, dan lain sebagainya. Proses berikutnya tekan F5, maka akan muncul seperti Gambar 3.

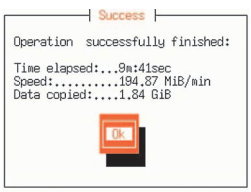
Gambar 3. Description Backup.

7. Deskripsi ini berguna untuk memberi gambaran apa isi dari image file yang dibuat. Boleh juga dikosongkan, kalau tidak berminat mengisinya. Selanjutnya, Partimage akan memberi informasi backup yang akan kita buat seperti Gambar 4.



Gambar 4. Information backup.

8. Setelah tekan *ok*, maka proses penyimpanan partition ke bentuk file akan berlangsung beberapa menit. Lama waktu ini tergantung dari seberapa besar ukuran data yang digunakan oleh partisi.
9. Pada contoh pembuatan ini, partisi linux yang digunakan berukuran 5 GB, dengan ukuran byte yang terpakai 1,84 GB. Ternyata ukuran image file adalah 758 MB, dengan waktu yang dibutuhkan untuk mem-backup 9 menit, 41 detik, dan untuk me-restore 5 menit, 15 detik, seperti terlihat di Gambar 5.



Gambar 5. Backup selesai.

10. File hasil pembuatan backup ini memiliki ekstensi .000 (.001, .002 dst kalau di-*split*).
11. Untuk mem-backup sistem operasi lain, caranya sama, namun hal yang perlu diperhatikan adalah partisi mana yang

mau di-backup, dan di mana image file akan disimpan.

Restore dengan Partimage

1. umount partisi yang akan di-restore. Setelah sistem optimal, partisi sistem di-umount untuk di-restore. Kalau me-restore sistem Linux, manfaatkan Linux live CD. Kalau me-restore Windows, bisa dilakukan melalui Linux yang sudah terinstal di harddisk, atau melalui Linux live CD.

```
# umount /dev/hda7
```

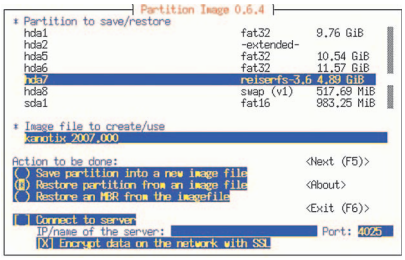
2. mount partisi yang di dalamnya ada image file, dan buka partisi. Setelah itu, disiapkan partisi yang akan digunakan untuk menyimpan image file. Misalnya:

```
# mount /dev/hda5 /media/hda5
# cd /media/hda5
```

3. Aktifkan Partimage:

```
# partimage
```

Pada layar akan muncul dialog seperti Gambar 6.

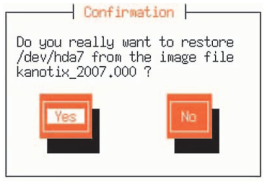


Gambar 6. Restore.

4. Pilih partisi hda7, dan image file yang digunakan adalah kanotix_2007.000. Aksi yang dipilih adalah *restore partition from an image file*. Setelah dipilih *Next*, maka akan muncul seperti Gambar 7.

Gambar 7. Description Restore.

5. Proses Partimage ini akan menunjukkan deskripsi yang dulunya kita tulis, sehingga kita dengan mudah bisa mengenali image file yang akan di-restore. Setelah menyampaikan deskripsinya, Partimage akan memperlihatkan *option restore*. Di sini bisa dipilih beberapa opsi, misalnya setelah selesai apa yang akan dilakukan (*Wait, Halt, Reboot* atau *Quit*).
6. Setelah *Ok*, maka akan muncul konfirmasi (Gambar 8) apakah yakin akan me-restore atau tidak, setelah dipilih *Yes*, maka proses restore akan segera dimulai.



Gambar 8. Really restore.

7. Setelah memilih Wait, Partimage akan menampilkan pesan berapa data yang di-*copy*, kecepatan serta waktu yang dibutuhkan, seperti dalam Gambar 9. Setelah *Ok*, maka proses restore selesai.

Gambar 9. Finish restore.

Cara cepat mem-backup dan restore dengan Partimage

Sebenarnya ada cara lain untuk mem-backup dan me-restore dengan cepat, yaitu dengan langkah-langkah berikut ini:

1. Lepaskan (umount) partisi yang akan di-restore atau di-backup:

```
# umount /dev/hda7
```

2. Setelah itu, siapkan partisi image file. Misalnya:

```
# mount /dev/hda5 /media/hda5
```

3. Lalu buka partisi tersebut:

```
# cd /media/hda5
```

4. Untuk mem-backup, lakukan langkah 1,2,3, lalu gunakan perintah:

```
# ./partimage -z1 -o -d save
/dev/hda7 kanotix_2007
```

5. Untuk me-restore, lakukan *step* 1,2,3, lalu tulislah:

```
# ./partimage restore /dev/hda7
kanotix_2007.000
```

Kesimpulan dan saran

1. Usahakan hanya menginstal software yang sudah mengenali semua hardware dengan baik, dan merupakan software yang sering kita pakai, sehinggga optimal (bisa di-restore secara efisien).
2. Jika Anda paham tentang *temporer* file, hapus dulu sehinggga diharapkan ukuran image file lebih kecil.
   a. Misalnya, backup Windows Anda bisa menghapus pagefile.sys, dan hibernat.sys pada direktori C:, kemudian cari *temp* file lainnya.
   b. Biasanya, aplikasi Windows sebelum menginstal, menaruh file sementara ke direktori temp. Direktori temp kadang dihapus, namun kadang tidak dihapus, sehingga kita perlu menghapusnya.
   c. Pada Linux, coba lihat direktori dan file temporari pada direktori *home* masing-masing user.
3. Simpan hasil backup pada partisi yang bebas. Misalnya, Anda memakai partisi 1 berisi Windows, partisi 2 berisi data, partisi 3 berisi master, partisi 5 berisi Linux, partisi 6 berisi *swap*, maka gunakan partisi 2 atau 3 sehinggga pada saat Windows Anda *crash*, image file Anda bisa segera digunakan. Demikian juga bila misalnya Linux crash, file backup Anda juga bisa segera digunakan.
4. Jika harddisk komputer sudah terlalu penuh, letakkan file backup pada CD atau pada DVD.
5. Untuk membakar CD/DVD, bisa dilakukan melalui Linux (menggunakan aplikasi k3b), ataupun melalui OS lainnya. Pada Windows, bisa menggunakan aplikasi Nero (paling umum digunakan pada Windows, cara penggunaannya hampir sama dengan k3b pada Linux).
6. File hasil pem-backup-an dapat digunakan melalui jaringan (proses backup dan proses restore). Hal ini sangat penting, apabila:
   a. Jumlah komputer yang dikelola banyak, pada suatu jaringan yang besar, misalnya laboratorium, perusahaan, pendidikan, dan lainnya.
   b. Jenis sistem operasi yang digunakan bervariasi (Linux, Windows, MacOS, dan lain-lain) pada suatu jaringan, sehinggga akan membantu *administrator* dalam pengelolaan jaringan.
7. Segeralah, dan jadwalkan backup pada komputer Anda, sehingga tidak terlalu banyak memikirkan yang tidak perlu. Beberapa teman penulis memiliki masalah berkaitan banyaknya virus komputer yang menyerang Windows, sehingga banyak waktu dan tenaga yang terkuras dalam menangani virus. Mau tidak mau, akan perlu pem-back-up-an komputer secara baik.

Solekhan [lekhan_sql@yahoo.com]

Berbicara Langsung dengan Server

Seorang *system administrator* yang bertugas menangani *server*, suatu saat pasti akan menemukan suatu masalah yang berkenaan dengan aplikasi server-nya. Dengan menggunakan program telnet, sejumlah permasalahan yang terjadi di server dapat segera ditelusuri, untuk kemudian dicari jalan pemecahannya.

Kadang-kadang dalam melakukan penyelesaian masalah (*troubleshooting*) sebuah *service*, kita perlu mengetes langsung dengan berkomunikasi dengan server, tanpa bantuan program klien, untuk mengetahui pesan kesalahan persis dari server. Contohnya, jika gagal mengambil *e-mail via* POP atau mengirim e-mail via SMTP, banyak program klien e-mail, termasuk Mozilla Thunderbird, tidak melaporkan pesan kesalahan yang cukup detail. Malah kadang-kadang program klien tertentu, atas nama "user friendliness", hanya melaporkan pesan kesalahan standar seperti ini: "Telah terjadi sebuah kesalahan. Silakan coba beberapa saat lagi, atau hubungi *admin* Anda." Kesalahan seperti apa? Di mana? Penyebabnya apa? Bagaimana cara memperbaikinya, selain mencoba lagi atau menghubungi admin? Sebab, sayalah admin-nya!

Untunglah, protokol-protokol Internet sebagian besar sifatnya teks, dan terdokumentasi dengan baik lewat dokumen-dokumen bernama RFC. Untuk berkomunikasi langsung dengan sebuah service Internet, seringkali kita hanya membutuhkan program telnet saja dan langsung mengetikkan dengan tangan, perintah-perintah dalam bahasa protokol yang sesuai.

Artikel ini menjelaskan beberapa protokol populer, disertai cara pemakaian umumnya.

Contents:

1. Menggunakan program telnet.
2. Troubleshooting program telnet.
3. Berbicara dengan web server.
4. Berbicara dengan mail server POP.
5. Berbicara dengan mail server SMTP.
6. Berbicara dengan server mail IMAP.

1. Menggunakan program telnet

Program telnet digunakan untuk terhubung ke sebuah mesin/server tertentu dan ke port tertentu, lalu kita dapat mengirimkan perintah dengan mengetikkan langsung, dan kita juga akan mendapatkan respon *output* dari si server. Dengan kata lain, kita berinteraksi langsung dengan sebuah server. Setelah terhubung, kita lalu perlu "berbahasa" protokol yang sesuai dengan aplikasi server yang kita hubungi. Misalnya jika kita menghubungi port 80, maka kemungkinan besar kita harus berbicara protokol HTTP. Jika ke port 25, maka protokol SMTP, dan seterusnya.

Catatan: jangan dibingungkan antara protokol Internet yang disebutkan di artikel ini dengan "protokol telnet" (port 23), yaitu yang digunakan untuk *login* ke *shell* sebuah mesin. Protokol ini telah amat tua, dan ditinggalkan karena tidak aman, dan kini biasanya diganti dengan protokol SSH (port 22).

Program telnet sendiri adalah program generik untuk terhubung port mana saja, dan *host* mana saja.

Di Unix/Linux, untuk menjalankan telnet:

```
$ telnet HOSTNAME_ATAU_IP PORT
```

Contoh:

```
$ telnet tokoku.com 80
```

Untuk keluar dari sesi telnet, tekan tombol *Ctrl-]* (tahan *Control,* setelah itu tekan tombol berlabel kurung siku tutup), lalu tekan *Enter*. Anda akan kembali ke *prompt* telnet. Tekan *Ctrl-D,* atau ketik *quit* diikuti *Enter*.

Di Windows juga tersedia program telnet. Anda dapat menjalankannya dari *Start Menu* > *Run...* lalu ketik telnet diikuti Enter. Atau bukalah terlebih dahulu *Command Prompt*, lalu ketik *telnet*, diikuti Enter.

Default-nya *localecho* di program telnet Windows dimatikan. Jadi, jika kita mengetik perintah ke sebuah server, maka kita sendiri tidak melihat apa yang kita ketikkan. Maka kita harus menyalakannya dulu, agar bisa lebih mudah bekerja.

Setelah mengetik telnet, ketikkan:

```
set localecho
```

Barulah terhubung ke server yang diinginkan dengan perintah *o*, misalnya (perhatikan urutan: huruf "o" kecil, spasi, *hostname*, spasi, port):

```
o tokoku.com 80
```

Sama seperti di Unix/Linux, untuk keluar dari sesi telnet, tekan tombol Ctrl-] (tahan Control, setelah itu tekan tombol berlabel kurung siku tutup), lalu tekan Enter. Anda akan kembali ke prompt telnet. Tekan *Ctrl-Z,* atau ketik *quit,* diikuti Enter.

2. Troubleshooting program telnet

Jika Anda menerima pesan kesalahan "Connection refused", artinya di server tidak ada daemon/program/service yang *listen* di port tersebut. Atau bisa juga karena firewall di sisi Anda, atau di sisi server.

Jika saat ingin terhubung, hasilnya

diam saja atau *timeout*, bisa jadi karena koneksi Anda lambat, atau ada firewall di sisi Anda/server.

3. Berbicara dengan web server

Oke, setelah mulai familiar dengan program telnet, mari kita mencoba berbicara dengan service paling populer di seantero jagad Internet, yaitu *web* yang menggunakan protokol HTTP (*Hyper Text Transfer Protocol*). Untuk berbicara langsung dengan server HTTP, kita dapat menggunakan program telnet, dan terkoneksi ke port HTTP (default-nya 80, tapi dapat berbeda).

Sekilas protokol HTTP. Klien mengirimkan HTTP *request* berupa: 1 baris request, diikuti dengan nol atau lebih baris *header*, diakhiri dengan baris kosong (dan dapat diikuti informasi tambahan). Server akan mengirimkan HTTP *response* berupa 1 baris respon, diikuti nol atau lebih baris header, diikuti bodi respon.

Sintaks baris request:

```
METODE URI PROTOKOL
```

METODE misalnya GET, HEAD, POST, dll. URI misalnya "/" (untuk meminta halaman utama), "/path/file.html?id=123", dll. PROTOKOL dapat berupa HTTP/1.0 atau HTTP/1.1.

Contoh baris request:

```
GET / HTTP/1.0
```

Contoh lain:

```
HEAD / HTTP/1.0
```

Sintaks header:

```
NAMA_HEADER: NILAI_HEADER
```

Contoh:

```
User-Agent: Mozilla/1.0
```

Contoh lain:

```
Pragma: no-cache
```

Sintaks baris respon:

```
PROTOKOL KODE_RESPON KETERANGAN
```

PROTOKOL adalah HTTP/1.0 atau HTTP/1.1. KODE_RESPON adalah angka 3 digit, 2xx artinya berhasil, 3xx artinya *redirection*, 4xx artinya ada kesalahan di sisi klien, 5xx artinya ada kesalahan di sisi server. Kode-kode umum adalah 200 (OK), 302, 303, 401, 403 (*Forbidden*), 404 (*Not Found*), 500 (*Internal server error*). Kode lengkap dapat dilihat di RFC.

Contoh:

```
HTTP/1.0 200 OK
```

Contoh lain:

```
HTTP/1.1 404 Not Found
```

Berikut ini contoh lengkap sebuah sesi HTTP. Yang dicetak tebal adalah yang kita ketikkan.

Meminta halaman *http://www.yahoo.com/*:

```
$ telnet www.yahoo.com 80
Trying www.yahoo.com...
Connected to www.yahoo.com.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 200 OK
Date: Fri, 16 Mar 2007 04:24:16 GMT
P3P: policyref="http://p3p.yahoo.
com/w3c/p3p.xml", CP="CAO DSP COR
CUR ADM DEV TAI PSA PSD IVAi IVDi
CONi TELo OTPi OUR DELi SAMi OTRi
NRi PUBi IND PHY ONL UNI PUR FIN COM
NAV INT DEM CNT STA POL HEA PRE
GOV"
Content-Type: text/html
X-Cache: MISS from server.
localdomain
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML...
```

Tip: Untuk mengintip *traffic* HTTP, Anda bisa pula menggunakan fitur di browser, seperti misalnya menggunakan *plugin* HTTPWatch di Internet Explorer, atau *add-on* seperti Firebug di Firefox. Di wget ada opsi -S ("server response"), untuk melihat header yang dikembalikan server. Di curl ada opsi -D ("dump headers").

Spesifikasi protokol HTTP ada di RFC 2616.

4. Berbicara dengan mail server POP

Protokol POP3 (*Post Office Protocol* version 3) adalah protokol untuk mengambil e-mail dari server. Untuk berbicara langsung dengan server POP3, kita dapat menggunakan program telnet, dan terkoneksi ke port POP3 (default-nya 110, tapi dapat berbeda).

Sekilas protokol POP3. Pertama, Anda login dengan memasukkan username dan password:

```
USER username
PASS pass
```

Lalu Anda dapat mendaftar jumlah mail yang ada di *mailbox*, dengan perintah LIST:

```
LIST
```

Output-nya adalah daftar nomor urut dan ID message.

Untuk mengambil sebuah e-mail digunakan perintah RETR ("retrieve"):

```
RETR nomor_urut_message
```

Untuk menghapus, dapat digunakan perintah DELE ("delete"):

```
DELE nomor_urut_message
```

Daftar perintah selengkapnya dapat dilihat di RFC.

Berikut ini sebuah contoh sesi POP3. Yang dicetak tebal adalah yang kita ketikkan.

Login yang berhasil:

```
$ telnet localhost 110
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK Hello there.
USER steven@steven.builder.
localdomain
+OK Password required.
PASS xxx
+OK logged in.
```

Login yang gagal:

```
USER steven@steven.builder.
localdomain
+OK Password required.
PASS xxxxx
-ERR Login failed.
```

Melihat e-mail yang ada di mailbox:

```
LIST
+OK POP3 clients that break here,
they violate STD53.
1 3273
2 14047
.
```

Mengambil sebuah e-mail:

```
RETR 1
+OK 3273 octets follow.
Delivered-To: xxxxxxxxxxxxxxxxxxxxx
Received: (qmail 20932 invoked from
network); 16 Mar 2007 02:27:18 -0000
Received: from unknown (HELO
xxxxxxxxxx ...
...(dst)...
.
```

Menghapus sebuah *message*:

```
DELE 1
+OK Deleted.
```

Keluar:

```
QUIT
+OK Bye-bye.
Connection closed by foreign host.
```

Silakan baca RFC 1989 untuk lebih detail.

5. Berbicara dengan mail server SMTP

Protokol SMTP (*Simple Mail Transfer Protocol*) digunakan untuk mengirim e-mail. Untuk berbicara langsung dengan server SMTP, kita dapat menggunakan program telnet, dan terkoneksi ke port SMTP (default-nya 25, tapi dapat berbeda). Ada juga port 587 (*submission*), yang juga digunakan untuk mengirim e-mail.

Catatan: jika ingin mengirim e-mail ntuk *domain*, katakanlah *yahoo.com*, terlebih dahulu kita harus mencari tahu MX record untuk yahoo.com tersebut, misalnya:

```
$ host -t mx yahoo.com
yahoo.com mail is handled by 1 d.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 e.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 f.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 g.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 a.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 b.mx.
mail.yahoo.com.
yahoo.com mail is handled by 1 c.mx.
mail.yahoo.com.
```

Maka kita dapat mencoba menghubungi salah satu MX mis:

```
$ telnet d.mx.mail.yahoo.com 25
```

Kita tidak menghubungi langsung yahoo.com di port 25, karena belum tentu MX untuk domain yahoo.com itu ada di mesin yang sama dengan yahoo.com.

Sekilas protokol SMTP. Setelah terkoneksi, server akan memberikan baris *banner* (*welcome message*). Klien mengirimkan baris perintah. Server mengirimkan baris respon berupa kode hasil 3 digit, diikuti keterangan. Kode 2xx artinya berhasil, kode 4xx artinya terjadi kesalahan temporer dan klien dipersilakan mencoba lagi beberapa saat mendatang, kode 5xx artinya terjadi kesalahan permanen.

Contoh, kita ingin mengirim e-mail ke *steven@serverku.com*. Pertama, kita mencari tahu MX untuk domain ini:

```
$ host -t mx serverku.com
serverku.com mail is handled by 10
mail.serverku.com.

$ telnet mail.serverku.com 25
Trying 203.130.198.32...
Connected to mail.serverku.com.
Escape character is '^]'.
220 serverku.com ESMTP ready
(Spanel-SMTPD 0.4.5-sp1.3)
```

Lalu, kita sebutkan siapa pengirimnya:

```
MAIL FROM:<pengirim@domain.com>
250 OK(250)
```

Jika tidak ingin menyebutkan pengirimnya (*Return-Path* kosong), kita bisa menuliskan:

```
MAIL FROM:<>
```

Lalu kita sebutkan penerimanya:

```
RCPT TO:<steven@serverku.com>
250 OK(250)
```

Baris RCPT ini dapat diulang untuk penerima-penerima yang lainnya jika ada, misalnya kita ingin mengirim e-mail ini ke-3 orang lainnya di domain yang sama:

```
RCPT TO:<user2@serverku.com>
RCPT TO:<user2@serverku.com>
...
```

Catatan: jika domain-nya berbeda, misalnya *budi@yahoo.com*, maka tentu kita harus menghubungi MX yahoo.com, dan bukan di serverku.com.

Lalu kirimkanlah bodi e-mail-nya. Akhiri dengan sebuah baris berisi titik:

```
DATA
354 GO_AHEAD(354)
Subject: test
Ini hanya sebuah tes, harap abaikan
.
250 QUEUED(250) - e5326248c623f66e81
05345a3e3135dd
```

Untuk keluar:

```
QUIT
221 BYE(221)
Connection closed by foreign host.
```

Contoh lengkap sebuah sesi SMTP, mengirim e-mail dan sukses:

```
$ telnet serverku.com 25
Trying 203.130.198.32...
Connected to serverku.com.
Escape character is '^]'.
220 serverku.com ESMTP ready
(Spanel-SMTPD 0.4.5-sp1.3)
MAIL FROM:<pengirim@domain.com>
250 OK(250)
RCPT TO:<steven@serverku.com>
250 OK(250)
DATA
354 GO_AHEAD(354)
Subject: test

Ini hanya sebuah tes, harap abaikan
```

```
.
250 QUEUED(250) - fd641de480641ec3c3
fb0eae5d2deedd
QUIT
221 BYE(221)
Connection closed by foreign host.
```

Contoh mengirim e-mail yang ditolak, karena user tidak dikenali:

```
RCPT TO:<xxx@serverku.com>
550 ERR_UNKNOWN_USER(550) - I
couldn't find that mail user here
```

Otentikasi. Kadang untuk mengirim secara *relay* (mengirim ke domain lain), sebuah server membutuhkan otentikasi. Protokol otentikasi adalah sbb:

AUTH LOGIN

- Server akan merespon, minta username. Ketikkan-username-yang-diencode-menggunakan-base64
- Server akan merespon minta password, ketikkan-password-yang-di-encode-menggunakan-base64
- Server akan memberitahu *auth* kita benar atau salah

Untuk meng-encode base64, dapat digunakan skrip berikut:

```
$ perl -MMIME::Base64 -e'print
encode_base64("1234")'
MTIzNA==
```

Artinya, '1234' jika di-encode akan menjadi 'MTIzNA=='.

Contoh SMTP authentication: (misalnya username-nya adalah 'steven@serverku.com', dan password-nya adalah '1234', yang jika di-encode akan menjadi 'c3RldmVuQHNlcnZlcmt1LmNvbQ==' dan 'MTIzNA==').

```
$ telnet serverku.com 25
Trying 203.130.198.32...
Connected to serverku.com.
Escape character is '^]'.
220 serverku.com ESMTP ready
(Spanel-SMTPD 0.4.5-sp1.3)
AUTH LOGIN
334 VXNlcm5hbWU6
c3RldmVuQHNlcnZlcmt1LmNvbQ==
334 UGFzc3dvcmQ6
MTIzNA==
```

Spesifikasi SMTP ada di RFC 2821.

6. Berbicara dengan server mail IMAP

Protokol IMAP (*Internet Message Access Protocol*) digunakan untuk mengakses dan memanipulasi pesan, dan mailbox mail di server. Untuk berbicara langsung dengan server IMAP, kita dapat menggunakan program telnet, dan terkoneksi ke port IMAP (default-nya 143, tapi dapat berbeda).

Sekilas protokol IMAP. Setiap baris request dari klien perlu di-*prefiks* (diawali) dengan sebuah *string* penanda. Pada contoh-contoh di artikel ini digunakan a001.

Login:

```
$ telnet mail.host.com 143
Trying 1.2.3.4...
Connected to mail.host.com
Escape character is '^]'.
* OK [CAPABILITY IMAP4REV1 LITERAL+
SASL-IR LOGIN-REFERRALS AUTH=LOGIN]
mail.host.com IMAP4rev1 2004.350
at Fri, 16 Mar 2007 13:51:21 -0500
(CDT)
a01 LOGIN someuser somepass
RESPONSE: a001 OK User logged in
```

Mengetahui *capability* IMAP service:

```
a001 CAPABILITY
* CAPABILITY IMAP4REV1 LITERAL+ IDLE
NAMESPACE MAILBOX-REFERRALS BINARY
UNSELECT SCAN SORT THREAD=REFERENCES
THREAD=ORDEREDSUBJECT MULTIAPPEND
SASL-IR LOGIN-REFERRALS AUTH=LOGIN
a001 OK CAPABILITY completed
```

Logout:

```
a001 LOGOUT
* BYE mail.host.com IMAP4rev1 server
terminating connection
a002 OK LOGOUT completed
Connection closed by foreign host.
```

Melihat daftar mailbox yang ada (pada contoh, semua anak Inbox):

```
a001 LIST "Inbox" "*"
* LIST (\HasNoChildren) "." "INBOX.
omail.log.old"
* LIST (\HasChildren) "." "INBOX.
omail.log"
* LIST (\HasNoChildren) "." "INBOX.
Trash"
* LIST (\HasNoChildren) "." "INBOX.
Spam"
* LIST (\HasNoChildren) "." "INBOX.
Sent"
* LIST (\HasNoChildren) "." "INBOX.
Drafts"
* LIST (\Noselect \HasChildren) "."
"INBOX.omail"
a001 OK LIST completed
```

Men-*select* sebuah mailbox (pada contoh, bernama INBOX):

```
a001 SELECT INBOX
* FLAGS (\Answered \Flagged \Draft
\Deleted \Seen)
* OK [PERMANENTFLAGS (\Answered \
Flagged \Draft \Deleted \Seen \*)]
* 1242 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1062186210]
* OK [UIDNEXT 1246]
a03 OK [READ-WRITE] Completed
```

Mendaftar semua message yang ada pada sebuah mailbox yang sedang di-select:

```
a001 FETCH 1:* FLAGS
* 1 FETCH (FLAGS (\Seen))
* 2 FETCH (FLAGS (\Seen))
* 3 FETCH (FLAGS (\Seen))
* 4 FETCH (FLAGS (\Seen))
* 5 FETCH (FLAGS (\Seen))
* 6 FETCH (FLAGS (\Seen \Answered))
...
a001 OK FETCH completed
```

Mengambil header-header sebuah message (pada contoh, urutan pertama dan mengambil full headers).

```
a001 FETCH 1 full
* 1 FETCH (FLAGS (\Seen)
INTERNALDATE "15-Mar-2000 13:10:14
-0500" RFC822.SIZE 1
553 ENVELOPE ("Wed, 15 Mar 2007
13:10:11 -0600" "Perl Stuff"
(("Rusty Nejdl" NIL
"rnejdl" "verio.net")) (("Rusty
Nejdl" NIL "rnejdl" "verio.net"))
(("Rusty Nejd
l" NIL "rnejdl" "verio.net")) ((NIL
NIL "rnejdl" "verio.net")) ((NIL NIL
"ttodd"
"verio.net")) NIL NIL
"<20000315131011.A5381@tethys.
ringofsaturn.com>") BODY ("
TEXT" "PLAIN" ("CHARSET" "us-ascii")
NIL NIL "7BIT" 359 9))
a001 OK FETCH completed
```

Mengambil bodi message:

```
a001 FETCH 1 body[text]
```

Set quota:

```
a001 SETQUOTA "" (STORAGE 512)
```

Get quota:

```
a001 GETQUOTA ""
* QUOTA "" (STORAGE 10 512)
a001 OK Getquota completed
```

Get quota root:

```
a001 GETQUOTAROOT "Inbox"
```

Spesifikasi IMAP ada di RFC 3501.

Steven Haryanto [steven@masterwebnet.com]

Membangun Paket RPM, DPKG dan TGZ dengan CheckInstall

Di tulisan ini, kita akan membahas pembuatan paket RPM, DPKG ataupun TGZ Slackware dengan mudah dan cepat menggunakan program CheckInstall. Dengan demikian, Anda bisa memaketkan program yang Anda bangun untuk berbagai target distribusi, tanpa harus repot-repot membangun secara manual.

Seperti kita ketahui bersama, berbagai distribusi Linux umumnya menggunakan sistem *package management* tertentu, untuk memudahkan pengaturan *software*. Package management tersebut umumnya mencakup instalasi, penghapusan, *upgrade* dan *query* paket/isi paket. Tergantung dari masing-masing distribusi Linux, package management yang digunakan bisa sangat sederhana atau sangat rumit.

Di dunia *open source*, terdapat banyak package management. Karakteristik mereka berbeda-beda. Ada yang sangat kaya fitur. Ada yang super sederhana. Ada yang tidak menuruti standar, dan lain sebagainya. Beberapa diantaranya sangat populer, karena digunakan pada distribusi Linux yang banyak digunakan.

Berikut ini adalah tiga package management populer yang umum digunakan oleh berbagai distribusi Linux:

- RPM (RPM Package Manager): dikembangkan oleh Red Hat, saat ini digunakan di banyak distribusi Linux, seperti RHEL, Fedora, openSUSE, Mandriva, dan lain sebagainya. File paket memiliki ekstensi .rpm.
- DPKG: dikembangkan oleh proyek Debian, saat ini juga digunakan oleh banyak distribusi Linux (sebagian besar oleh distribusi berbasis Debian). Terkenal sebagai package management yang pertama memiliki kemampuan resolusi dependensi (APT). File paket memiliki ekstensi .deb.
- TGZ Slackware: digunakan oleh distribusi Slackware. Merupakan package management yang relatif sederhana, namun tetap *powerful*. File paket memiliki ekstensi .tgz.

Secara sederhana, dalam satu file paket, umumnya terdiri dari file-file anggota paket, berbagai *metadata* dan *script-script* pembantu yang dipaketkan menjadi satu, dan umumnya dikompres. *Tool-tool* package management kemudian digunakan untuk memanipulasi file paket tersebut, dan lebih jauh lagi, memanipulasi file-file hasil instalasi.

Tidak ada yang sangat rahasia dengan file paket tersebut. Semua tool yang digunakan untuk membangun paket tersedia, dan hampir semuanya merupakan tool standar.

Namun, membangun paket juga bukan pekerjaan yang sederhana, yang bisa dikerjakan sambil lalu, seperti halnya mengompres satu direktori. Tinggal berikan perintah, tunggu, dan selesai. Selain masalah kompres-mengompres, kita juga harus mengurus berbagai metadata yang umum dimiliki oleh suatu file paket, berbagai *script,* dan hal-hal lainnya.

Bagi yang lebih suka menggunakan alat bantu yang bisa diandalkan, CheckInstall akan sangat membantu pembuatan paket. Kita bisa membangun paket Debian, Slackware dan RPM dengan mudah, dan cepat.

CheckInstall

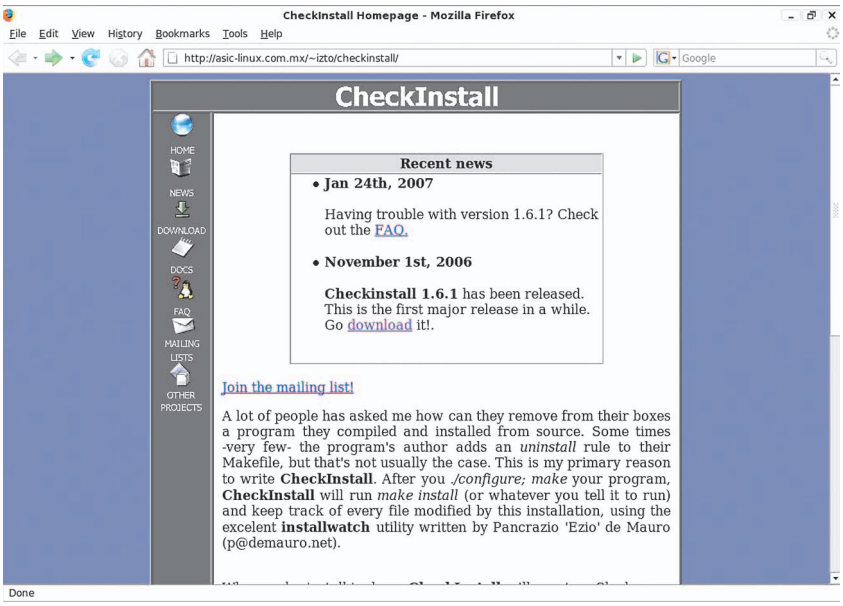
Umumnya, langkah-langkah instalasi program dari *source code* adalah dengan menjalankan serangkaian perintah berikut:

- Konfigurasi dengan script *configure*.
- Melakukan kompilasi dengan bantuan *makefile*, dengan menjalankan program make.
- Melakukan instalasi dengan bantuan makefile, dengan memberikan perintah make *install*, atau yang sejenis.

Ketika melakukan konfigurasi dan kompilasi, file-file yang dihasilkan umumnya masih berada di dalam *tree* direktori source code. Namun, begitu kita menjalankan make install, maka file-file akan segera menyebar di filesistem.

Bagaimana kalau kita ingin melakukan *uninstall*? Beberapa menyediakan fasilitas/*rule* untuk uninstall dengan perintah make uninstall di tree source code. Namun, tidak semuanya menyediakan fasilitas ini. Sistem pun berpotensi kotor, dan kita harus melongok ke dalam makefile, dan kemudian melakukan penghapusan secara manual.

Dengan menjalankan make install sebelumnya untuk instalasi program, selain masalah uninstall, proses upgrade juga tidaklah mudah untuk dilakukan. Tidak setiap versi datang dengan file-file yang sama. Ada kemungkinan penambahan atau pengurangan file, atau sistem peletakan yang berbeda-beda. Bahkan, ada



Situs web CheckInstall.

pula yang mengubah nama *executable*.

Dengan menggunakan CheckInstall, setelah Anda melakukan konfigurasi (dengan script configure), dan kompilasi (dengan make), janganlah langsung memberikan perintah make install. Sebaliknya, berikanlah perintah CheckInstall make install (atau cukup CheckInstall saja), yang akan membantu kita membangun paket, melacak file-file yang akan dikopikan dengan make install, dan membangun paket untuk kita, tanpa sistem harus telanjur menjadi kotor (dan harus dibersihkan secara manual).

Lebih jauh lagi, kita tidak harus terpaku pada perintah make install ketika menggunakan CheckInstall. Perintah lain yang sejenis pun didukung.

Program yang luar biasa ini dibangun oleh Felipe Eduardo Sánchez Díaz Durán. Versi terakhir pada tulisan ini dibuat adalah 1.6.1, yang dirilis pada 1 November 2006 yang lalu.

Instalasi CheckInstall sendiri dapat dilakukan dengan mudah, setelah sebelumnya kita men-*download* paket untuk package management distro yang kita gunakan. *Link* download bisa didapatkan di alamat *http://asic-linux.com.mx/~izto/CheckInstall/download.php*. Download-lah paket yang cocok untuk sistem Anda, dan lakukanlah instalasi seperti biasa:

- RPM, dengan rpm -i <file_paket>.
- DPKG, dengan dpkg -i <file_paket>.
- TGZ slackware, dengan installpkg <file_paket>.

Untuk informasi selengkapnya, kunjungilah *website* CheckInstall di *http://asic-linux.com.mx/~izto/checkinstall/download.php*.

Membangun paket lintas Package Management

CheckInstall memungkinkan kita untuk membangun paket untuk package management yang berbeda dengan yang digunakan, selama tool yang dibutuhkan tersedia. Dengan demikian, sebagai contoh, kita bisa membangun paket RPM di Debian.

Namun, perhatikan juga masalah pustaka sistem. Setiap sistem dapat menggunakan pustaka C ataupun pustaka lain yang benar-benar berbeda dengan sistem target. Untuk program yang relatif sederhana, atau kalau Anda ingin melakukan link secara statik, barangkali membangun paket untuk distro lain bisa digunakan. Kalau hal ini dilakukan, selalulah melakukan pengujian secara mendalam.

Penulis, di dalam tulisan ini, akan menggunakan pendekatan untuk membangun tanpa lintas package management. Dengan demikian, untuk paket Slackware, penulis akan bekerja di sistem berbasis Slackware. Untuk membangun paket RPM, penulis akan bekerja di distro berbasis RPM. Untuk membangun paket Debian, penulis akan bekerja di distro berbasis Debian.

Program contoh

Sebagai program contoh, penulis akan mempergunakan moc (*Music On Console*), sebuah audio player yang dapat berjalan di *text* console. Penulis akan mempergunakan versi 2.4.1.

Untuk download dan informasi selengkapnya, kunjungilah website-nya, *http://moc.daper.net/*.

Anda, tentu saja, silakan mempergunakan program lain yang diinginkan. Penggunaan moc di sini hanyalah contoh.

Lakukanlah langkah-langkah berikut ini untuk melakukan kompilasi moc.

- Ekstraksi arsip source code, dan masuk ke direktori hasil ekstrak:

```
$ tar jxvf moc-2.4.1.tar.bz2
$ cd moc-2.4.1
```

- Konfigurasi dengan *prefix /usr:*

```
$ ./configure -prefix=/usr
```

- Kompilasi:

```
$ make
```

Setelah ini, jangan lakukan instalasi. Kita akan mempergunakan bantuan CheckInstall.

Sebagai catatan, untuk setiap distro, penulis akan melakukan kompilasi moc. Di tulisan ini, proses kompilasi dibahas sekali saja, agar tidak ada redundansi.

Membangun paket Slackware

Untuk membangun paket Slackware, penulis menggunakan distribusi Zenwalk 4.4.1. Barangkali, memang akan lebih klop kalau menggunakan Slackware. Sebagai catatan, ketika menginstal CheckInstall, penulis menggunakan paket untuk Slackware, dan bisa diinstal, dan dipergunakan dengan baik.

Secara default, CheckInstall akan diinstal di */usr/local/sbin/CheckInstall*.

Untuk membangun paket, pastikan Anda telah melakukan langkah-langkah yang telah dibahas pada bagian Program contoh.

Berikanlah perintah berikut ini untuk membangun paket Slackware (sebagai *root*):

```
# checkinstall
...
...
The package documentation directory
./doc-pak does not exist.
Should I create a default set of
package docs?  [y]:
```

Untuk saat ini, jawablah dengan n. Kita akan membahas ini di bagian script, dokumentasi dan lainnya, di bagian akhir tulisan.

```
Please choose the packaging method
you want to use.
```

```
Slackware [S], RPM [R] or Debian
[D]? S
```

Pilihlah S untuk membangun paket Slackware.

```
Please write a description for the
package. Remember that pkgtool shows
only the first one when listing
packages so make that one
descriptive.

End your description with an empty
line or EOF.
```

Masukkan deskripsi paket (akhiri dengan baris kosong). Contoh:

```
>> moc - music on console
>> console audio player for Linux/
UNIX
>> more information: http://moc.
daper.net
>>
...
...
1 -  Summary: [ moc - music on
console ]
2 -  Name:    [ moc ]
3 -  Version: [ 2.4.1 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ moc-2.4.1 ]
9 -  Alternate source location: [  ]
10 - Requires: [  ]

Enter a number to change any of them
or press ENTER to continue:
```

Pilihlah nomor untuk informasi yang ingin Anda ubah. Penulis akan mengubah nomor:

- 6, menjadi Applications/Multimedia.
- 7, menjadi i486.

Setelah kita meneruskan ke langkah berikutnya, layar akan dipenuhi dengan *output* perintah untuk melakukan instalasi. Setelah selesai, sebuah paket otomatis akan terbentuk:

```
**********************************

 Done. The new package has been
saved to

 /home/nop/temp/program/Devel/
MULTIMEDIA/MOC/moc-2.4.1/moc-2.4.1-
i486-1.tgz
 You can install it in your system
anytime using:

     installpkg moc-2.4.1-i486-1.tgz

**********************************
```

Paket yang dihasilkan bisa diinstal dengan installpkg, seperti pesan yang ditampilkan oleh CheckInstall.

Contoh instalasi:

```
# installpkg moc-2.4.1-i486-1.tgz
```

Membangun paket Debian

Untuk membangun paket Debian, penulis menggunakan distribusi Debian 3.1. Secara default, CheckInstall akan diinstal di */usr/local/sbin/checkinstall*.

Untuk membangun paket, pastikan Anda telah melakukan langkah-langkah yang telah dibahas pada bagian Program contoh.

Berikanlah perintah berikut ini untuk membangun paket Debian (sebagai root):

```
# checkinstall
...
...
The package documentation directory
./doc-pak does not exist.
Should I create a default set of
package docs?  [y]: n
```

Sama seperti sebelumnya, untuk saat ini, jawablah dengan n. Kita akan membahas ini di bagian script, dokumentasi dan lainnya, di bagian akhir tulisan.

```
Please choose the packaging method
you want to use.
Slackware [S], RPM [R] or Debian
[D]? D
```

Pilihlah D untuk membangun paket Debian.

```
Please write a description for the
package.
End your description with an empty
line or EOF.
```

Masukkan deskripsi seperti contoh berikut (akhiri dengan baris kosong):

```
>> music on console
>> console audio player for Linux/
UNIX
>> more information: http://moc.
daper.net
>>

...
...
0 -  Maintainer: [ nop@nop1.
noprianto.com ]
1 -  Summary: [ music on console ]
2 -  Name:    [ moc ]
3 -  Version: [ 2.4.1 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
6 -  Group:   [ checkinstall ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ moc-2.4.1 ]
9 -  Alternate source location: [  ]
10 - Requires: [  ]

Enter a number to change any of them
or press ENTER to continue:
```

Pilihlah nomor untuk informasi yang ingin Anda ubah. Penulis akan mengubah nomor:

- 0, menjadi nama *maintainer*, contoh: Nama Maintainer <email. maintainer@server.com>
- 6, menjadi *sound*.

Lanjutkanlah proses pembuatan paket dengan menekan tombol ENTER. Setelah selesai, sebuah paket otomatis akan terbentuk:

```
**********************************

 Done. The new package has been
saved to

 /root/moc-2.4.1/moc_2.4.1-1_i386.
deb
 You can install it in your system
anytime using:

      dpkg -i moc_2.4.1-1_i386.deb

**********************************
```

Contoh instalasi:

```
# dpkg -i moc_2.4.1-1_i386.deb
Selecting previously deselected
package moc.
(Reading database ... 80825 files and
directories currently installed.)
Unpacking moc (from moc_2.4.1-1_
i386.deb) ...
Setting up moc (2.4.1-1) ...
```

Sebagai catatan, paket ini memang tidak *lintian-clean*, namun apabila diinginkan, kita bisa melakukan perubahan yang tidak terlalu rumit, untuk menjadikannya lintian-clean (tidak dibahas di tulisan ini; bacalah *New Maintainer's Guide*, dan dokumentasi terkait lainnya untuk informasi selengkapnya). Contoh

pemeriksaan lintian:

```
$ lintian moc_2.4.1-1_i386.deb
W: moc: executable-not-elf-or-script
./usr/lib/moc/decoder_plugins/
libmp3_decoder.la
W: moc: executable-not-elf-or-script
./usr/lib/moc/decoder_plugins/
libvorbis_decoder.la
W: moc: executable-not-elf-or-script
./usr/lib/moc/decoder_plugins/
libsndfile_formats_decoder.la
E: moc: no-copyright-file
W: moc: binary-has-unneeded-section
./usr/bin/mocp .comment
E: moc: debian-changelog-file-missing
```

Membangun paket RPM

Untuk membangun paket RPM, penulis menggunakan distribusi openSUSE 10.2. Secara default, CheckInstall akan diinstal di */usr/local/sbin/checkinstall*.

Untuk membangun paket, pastikan Anda telah melakukan angkah yang telah dibahas pada bagian Program contoh.

Berikanlah perintah berikut ini untuk membangun paket RPM (sebagai root):

```
# checkinstall
...
...
The package documentation directory
./doc-pak does not exist.
Should I create a default set of
package docs?  [y]: n
```

Seperti kedua contoh sebelumnya, untuk saat ini, jawablah dengan n. Kita akan membahas ini di bagian script, dokumentasi dan lainnya, di bagian akhir tulisan.

```
Please choose the packaging method
you want to use.
Slackware [S], RPM [R] or Debian
[D]? R
```

Pilihlah R untuk membangun paket RPM.

```
Please write a description for the
package.
End your description with an empty
line or EOF.
```

Masukkan deskripsi seperti contoh berikut (akhiri dengan baris kosong):

```
>> moc - music on console
>> console audio player for Linux/
UNIX
>> more information: http://moc.
daper.net
>>
...
...
1 -  Summary: [ moc - music on
console ]
2 -  Name:    [ moc ]
3 -  Version: [ 2.4.1 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ moc-2.4.1 ]
9 -  Alternate source location: [  ]
10 - Requires: [  ]
11 - Provides: [ moc ]

Enter a number to change any of them
or press ENTER to continue:
```

Pilihlah nomor untuk informasi yang ingin Anda ubah. Penulis akan mengubah nomor:

- 6, menjadi *Productivity/Multimedia/Sound*.
- 7, menjadi i586.

Lanjutkanlah proses pembuatan paket dengan menekan tombol ENTER. Setelah selesai, sebuah paket otomatis akan terbentuk:

```
**********************************

 Done. The new package has been
saved to

 /usr/src/packages/RPMS/i586/moc-
2.4.1-1.i586.rpm
 You can install it in your system
anytime using:

      rpm -i moc-2.4.1-1.i586.rpm

**********************************
```

Perhatikanlah bahwa paket disimpan di */usr/src/packages/RPMS/i586/*.

Contoh instalasi:

```
# rpm -iv /usr/src/packages/RPMS/
i586/moc-2.4.1-1.i586.rpm
```

Script, dokumentasi dan lainnya

Untuk mempermudah pembuatan paket, menambahkan script ataupun mengatur dokumentasi paket, beberapa hal bisa dilakukan.

description-pak

Simpanlah deskripsi paket di dalam file ini. Apabila file ini tidak ditemukan, maka CheckInstall akan menanyakan deskripsi paket. Apabila ditemukan, maka isi file ini akan digunakan sebagai *description* dan *summary*.

doc-pak

Secara default, apabila direktori *doc-pak* tidak ditemukan, maka CheckInstall akan menawarkan untuk membuatkannya untuk kita. Pada pembahasan-pembahasan sebelumnya, kita menolak sehingga paket tidak memiliki dokumentasi (ini sangat tergantung package management, dan software yang ingin dipaketkan).

Buatlah direktori doc-pak, dan simpanlah semua file dokumentasi di dalam direktori ini.

Berbagai script

Beberapa package management mendukung berbagai script yang dapat dijalankan pada beberapa kondisi:

- Sebelum paket diinstal.
- Setelah paket diinstal.
- Sebelum paket dihapus.
- Setelah paket dihapus.

Untuk package management yang mendukung, file-file berikut ini bisa dibuat:

- *preinstall-pak*, akan dijalankan sebelum paket diinstal.
- *postinstall-pak*, akan dijalankan setelah paket diinstal.
- *preremove-pak*, akan dijalankan sebelum paket dihapus.
- *postremove-pak*, akan dijalankan setelah paket dihapus.

Pada Slackware, kita mempergunakan script postinstall-pak, ataupun install-pak.

Untuk kustomisasi lainnya, CheckInstall bekerja dengan beberapa variabel ataupun opsi yang diberikan pada CheckInstall. Bacalah file README, yang tersimpan di */usr/doc/checkinstall/README* .

Bisa kita lihat menggunakan CheckInstall, kita bisa membangun paket untuk berbagai distribusi dengan mudah dan cepat. Memang, untuk paket yang yang benar-benar baik, kita masih perlu melakukan banyak hal. Tapi, secara umum, CheckInstall sudah sangat banyak membantu kita.

Sampai di sini dulu pembahasan kita. Selamat mencoba.

Noprianto [noprianto@infolinux.co.id]

Membangun init script Sederhana

Ketika sistem Linux di-*boot*, berbagai *init script* (atau *service* script) akan dijalankan untuk menjalankan berbagai aplikasi/servis yang dibutuhkan. Script-script tersebut selanjutnya dapat juga kita gunakan untuk mengontrol aplikasi/servis individual, ketika sistem sudah selesai di-boot.

Proses boot Linux sangatlah fleksibel, walau memang sangat tergantung distribusi yang digunakan. Sebagai contoh umum, pertama-tama, kernel akan di-*load* ke memory, dan *image initrd* atau *initramfs* akan dijalankan (initrd atau initramfs adalah opsi, walau umum digunakan). Setelah semua yang dibutuhkan tercapai (bisa dikerjakan pada sistem initrd atau initramfs), init di *root filesystem* yang sesungguhnya akan dijalankan. Umumnya, program init akan membaca file *konfigurasi /etc/inittab*. Selanjutnya, *system initialization* script yang didefinisikan pun akan menjalankan berbagai script init (sesuai *runlevel*), yang disebutkan pada awal tulisan ini.

Sebelum kita melanjutkan, sebagai contoh distribusi, kita akan menggunakan Debian dan Zenwalk (berbasis Slackware).

Runlevel sendiri, secara sederhana, bisa diartikan sebagai mode operasi sistem Linux. Dalam implementasinya, perbedaan secara mendasar antara mode satu dan mode lainnya adalah pada script-script yang dijalankan. Runlevel pada setiap distribusi bisa berbeda. Berikut ini adalah contoh runlevel pada berbagai distribusi:

- Debian: 0: sistem di-*halt*, 1: *single user*, 2,3,4,5: *multiuser* dengan *display manager*, 6: sistem di-*reboot*.
- Redhat dan turunannya: 0: sistem di-halt, 1: single user, 2: bisa dikustomisasi oleh user, 3: multiuser tanpa display manager, 4: bisa dikustomisasi oleh user, 5: multiuser dengan display manager, 6: sistem di-reboot.
- SUSE dan turunannya: 0: sistem di-halt, 1: single user, 2: multiuser tanpa jaringan, 3: multiuser tanpa display manager, 4: bisa dikustomisasi oleh user, 5: multiuser dengan display manager, 6: sistem di-reboot.
- Slackware: 0: sistem di-halt, 1: single user, 2,3: multiuser tanpa display manager, 4: multiuser dengan display manager, 5: multiuser tanpa display manager, 6: sistem di-reboot.

Selanjutnya, penanganan init script juga berbeda, tergantung dari "aliran" yang dianut. Sebagian besar distribusi Linux populer menggunakan aliran System V, beberapa menggunakan aliran BSD, dan beberapa yang lain menggunakan aliran sendiri.

Kekuatan aliran System V adalah fleksibel dan skalabel, sementara kelemahannya adalah relatif lebih kompleks. Kekuatan aliran BSD adalah kesederhanaan, namun di lain sisi, salah melakukan pengeditan bisa berakibat pada sistem yang tidak dapat di-boot dengan benar (walau hal ini bisa diantisipasi).

Contoh penganut aliran System V adalah Debian, Redhat dan SUSE. Contoh penganut aliran BSD adalah Slackware.

Pada dasarnya, aliran apapun yang dianut, baik System V, BSD ataupun aliran lain pada umumnya, script-script init tetap memegang peranan penting. Dengan penggunaan script-script, yang dijalankan pada saat boot sesuai runlevel, kita juga bisa menjalankan script tersebut, misalnya untuk menghentikan atau me-*restart* service, ketika sistem sedang berjalan.

Pengguna Linux barangkali cukup familiar dengan beberapa perintah berikut ini:

```
# /etc/init.d/apache2 restart
```

atau

```
# /etc/rc.d/rc.postgresql restart
```

Perintah-perintah tersebut kemudian akan menampilkan berbagai status, seperti *[OK], [FAILED], ok., done*, dan lain sebagainya. Kalau dijalankan tanpa argumen, umumnya script-script tersebut akan menampilkan argumen yang didukung.

Bicara lebih lanjut, script-script tersebut pada dasarnya adalah *shell* script biasa (bisa sederhana, bisa sangat kompleks), yang ketika diberikan argumen tertentu, akan melakukan tugas-tugas yang bersesuaian. Sebagai contoh:

- Ketika diberikan argumen *start*, script mungkin akan langsung menjalankan *daemon* tertentu atau menggunakan program pengatur daemon, yang datang bersama paket aplikasi untuk menjalankan daemon.
- Ketika diberikan argumen *stop*, script mungkin akan langsung menterminasi proses daemon atau menggunakan program pengatur daemon, yang datang bersama paket aplikasi untuk melakukan terminasi daemon.
- Ketika diberikan argumen restart, script mungkin akan menjalankan stop dan start, mengirim sinyal tertentu yang

dikenal oleh daemon, atau menggunakan program pengatur daemon yang datang bersama paket aplikasi untuk melakukan restart.

Tergantung aliran yang digunakan, pemanggilan script-script tersebut pada saat boot bisa berbeda pada saat sistem selesai di-boot (seperti cara yang disebutkan sebelumnya, yaitu dengan memanggil nama script disertai dengan argumen):

- Pada aliran system V, direktori */etc/rcX.d/* akan tersedia untuk masing-masing runlevel, di mana X adalah nomor runlevel. Di dalam direktori-direktori tersebut, berbagai *symlink* (ke script-script di */etc/init.d/*) dengan awal nama file S (untuk Start) atau K (untuk Kill), disertai nomor prioritas akan ditemukan.
- Pada aliran BSD, sebuah script untuk setiap runlevel akan memeriksa apakah script-script yang ingin dijalankan memiliki hak akses *executable*. Apabila memiliki hak akses tersebut, maka script-script tersebut akan dijalankan.

Lalu, bagaimana cara membangun init script tersebut? Kita bisa membuat sebuah shell script sederhana, *dummy*, seperti contoh berikut:

```
#!/bin/sh

start()
{
  echo -n "Starting dummy daemon..."
  sleep 2
  #real action
  if [ $? -eq 0 ]
  then
   echo " OK "
  else
   echo " FAILED "
  fi
}

stop()
{
  echo -n "Stopping dummy daemon..."
  sleep 2
  #real action
  if [ $? -eq 0 ]
  then
   echo " OK "
  else
   echo " FAILED "
  fi
}

restart()
{
  stop
  sleep 1
  start
}

case "$1" in
start)
  start
  ;;
stop)
  stop
  ;;
restart)
  restart
  ;;
*)
  echo "Usage: $0 <start | stop | restart>"
  exit 1
esac
exit 0
```

Berikanlah hak akses executable (kecuali Anda tidak menginginkannya, misalnya pada aliran BSD, dan membiarkan *admin* yang melakukannya, apabila dibutuhkan) dengan perintah:

```
$ chmod +x dummy
```

Setelah itu, kopikanlah ke lokasi init script, sesuai distro yang Anda gunakan:

- Debian: /etc/init.d/dummy.
- Berbasis Slackware: /etc/rc.d/rc.dummy.

Buatlah agar script tersebut dijalankan pada runlevel yang Anda inginkan:

- Debian: gunakan *tool* update-rc.d
- Berbasis Slackware: edit file runlevel yang diinginkan (misal: */etc/rc.d/rc.4*), dan tambahkan baris untuk menjalankan script */etc/rc.d/rc.dummy* (bisa juga dicek terlebih dahulu, apakah memiliki hak akses executable)

Gantilah bagian *#real action* pada script dengan aksi sesungguhnya (misal: menjalankan program tertentu, lengkap dengan argumen yang dibutuhkan). Di contoh ini, kita hanya menunda 2 detik, dengan program *sleep*.

Script yang kita buat sangatlah sederhana dan umum. Beberapa distro datang dengan fungsi atau program tertentu yang memudahkan kita. Beberapa juga datang dengan aturan yang harus dipatuhi. Berikut ini adalah beberapa catatan yang mungkin berguna:

- Umumnya, terdapat sebuah script yang mendefinisikan berbagai fungsi berguna, yang telah disediakan oleh distro. Kita umumnya perlu menjalankan script tersebut, di awal init script yang kita bangun.
- Beberapa distro senang menampilkan status OK dengan warna hijau, dan status gagal dengan warna merah. Apabila distro Anda menyediakan, Anda bisa menggunakannya (umumnya dengan hanya memanggil satu fungsi).
- Beberapa script menyediakan fasilitas status (disamping start, stop, restart), yang berguna untuk menampilkan status daemon (sedang berjalan dengan PID sekian, atau tidak berjalan).
- Beberapa script menyediakan fasilitas reload file konfigurasi, agar daemon membaca file konfigurasi yang baru, tanpa harus melakukan restart secara penuh. Ini, tentu saja sangat tergantung aplikasi yang kita proses.
- Beberapa distro mewajibkan pembuatan *lock* file pada saat service dijalankan, dan penghapusan lock file pada saat service dihentikan.
- Pada saat restart, kita mungkin perlu melakukan delay sebentar dengan program sleep. Ini sangat tergantung pada aplikasi. Pada contoh di tulisan ini, kita menunda selama 1 detik.
- Beberapa aplikasi, umumnya *database server*, harus membuat database *cluster* terlebih dahulu (hanya pada saat kali pertama), sebelum segala sesuatunya bisa dikerjakan. Periksalah apakah aplikasi yang Anda gunakan juga harus melakukan hal serupa.

Selalulah merujuk ke dokumentasi distro Anda. Materi yang kita bahas di tulisan ini sangatlah bergantung pada distro yang Anda gunakan. Hampir semua distro akan menyediakan dokumentasi akan hal ini. Secara mendasar, Anda bisa menggunakan contoh yang dibahas sebagai kerangka dari init script yang ingin Anda kembangkan.

Sampai di sini dulu pembahasan kita, selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

Bekerja dengan File dan Direktori di Linux

Di tulisan ini, kita akan membahas berbagai *system call* Linux dan fungsi standar untuk bekerja dengan file dan direktori. Dengan demikian, kita bisa memanipulasi file seperti membuat, menghapus, mengubah nama, mengubah mode, dan lain sebagainya. Kita juga bisa membuat, menghapus dan menampilkan isi direktori.

Bagi Anda yang membangun aplikasi sistem, bekerja dengan file dan direktori adalah hal yang umum dilakukan. Sebagai contoh, kita mungkin ingin menyimpan preferensi *user* ke file konfigurasi. Atau, barangkali kita perlu menghapus atau mengubah nama file-file tertentu. Untuk kebutuhan lebih lanjut, ada kalanya kita harus mengubah mode suatu file, untuk alasan keamanan misalnya.

Di Linux, tersedia semua system call yang dibutuhkan untuk bekerja dengan file (untuk berbagai tipe file, termasuk file reguler dan direktori). Kita juga bisa memanfaatkan fungsi standar yang tersedia.

Semua operasi file reguler dan direktori yang dibahas di tulisan ini, tentunya bisa Anda kombinasikan dengan pembahasan-pembahasan pemrograman lainnya, seperti user *interface* dengan *ncurses*. Misalnya, ketika Anda ingin membangun file manager sederhana berbasis ncurses.

Semua contoh di dalam tulisan ini di bangun di atas Zenwalk Linux 4.4.1, dengan gcc versi 3.4.6, namun seharusnya bisa diterapkan tanpa masalah pada sistem lainnya.

Membuat file

Pada pembahasan tentang pembuatan file, kita akan membahas dua contoh:

- 1.c: membuat file kosong, tanpa hak akses
- 1b.c: membuat beberapa file kosong dengan beberapa contoh hak akses

System call yang dipergunakan adalah open() dan close().

1.c

Berikut ini adalah *source code* 1.c:

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>

int main ()
{
  int fd;
  char f_name[256];

  strcpy (f_name, "1.test");

  fd = open (f_name, O_WRONLY |
O_CREAT);
  if (fd == -1)
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (-1);
  }
  else
  {
   fprintf (stdout, "File %s created
successfully with file descriptor
%d\n", f_name, fd);
  }

  close (fd);
  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 1 1.c
```

Contoh output:

```
$ ./1
File 1.test created successfully
with file descriptor 3
```

Penjelasan source code:

- Kita akan membuat file kosong dengan nama 1.test.
- Untuk membuka file (kalau tidak ditemukan, kita akan membuatnya), kita akan mempergunakan open()
  - Parameter pertama adalah nama file.
  - Parameter kedua adalah *flag*, di mana kita memberikan O_WRONLY (tulis saja), dan O_CREAT (apabila file tidak ditemukan, maka akan dibuat).
- open() akan mengembalikan nilai -1 apabila terjadi kesalahan, dan mengembalikan file *descriptor* baru apabila sukses.
- Untuk menutup file, kita mempergunakan close().
- Perhatikan bahwa file yang dibuat tidak memiliki hak akses untuk user, *group*, dan lainnya:

```
$ ls -al 1.test
---------- 1 nop users 0 2007-05-
12 13:48 1.test
```

Ketika dijalankan untuk kali kedua, program 1 akan menampilkan pesan kesalahan:

```
$ ./1
Permission denied
```

Hal tersebut disebabkan karena kita tidak bisa menimpa file 1.test tersebut, karena tidak memiliki hak.

Walau demikian, kita masih pemilik file tersebut, dan bisa menghapusnya dengan rm biasa:

```
$ rm -f 1.test
```

1b.c

Berikut ini adalah source code 1b.c:

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>

int main ()
{
  int fd1, fd2, fd3;

  fd1 = open ("1b.test.1", O_WRONLY
| O_CREAT, S_IRWXU );
  fd2 = open ("1b.test.2", O_WRONLY
| O_CREAT, S_IRUSR | S_IWUSR |
S_IRGRP );
  fd2 = open ("1b.test.3", O_WRONLY
| O_CREAT, S_IRUSR | S_IWUSR |
S_IRGRP | S_IROTH);

  close (fd1);
  close (fd2);
  close (fd3);

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 1b 1b.c
```

Penjelasan source code:

- Kita tidak menampilkan *output* apapun.
- Bacalah juga penjelasan untuk 1.c.
- Selain membuat file, kita juga memberikan mode tertentu:
  - 1b.test.1: user memiliki hak penuh (pada ls: -rwx------).
  - 1b.test.2: user memiliki hak rw, group memiliki hak r (pada ls: -rw-r-----).
  - 1b.test.3: user memiliki hak rw, group memiliki hak r, user lain memiliki hak r (pada ls: -rw-r—r--).

Berikut ini adalah mode-mode yang bisa digunakan:

- S_IRWXU : user memiliki hak baca, tulis, *execute*.
- S_IRUSR : user memiliki hak baca.
- S_IWUSR : user memiliki hak tulis.
- S_IXUSR : user memiliki hak execute
- S_IRWXG : group memiliki hak baca, tulis, execute.
- S_IRGRP : group memiliki hak baca.
- S_IWGRP : group memiliki hak tulis.
- S_IXGRP : group memiliki hak execute.
- S_IRWXO : user lain memiliki hak baca, tulis, execute.
- S_IROTH : user lain memiliki hak baca.
- S_IWOTH : user lain memiliki hak tulis.
- S_IXOTH : user lain memiliki hak execute.

Mode-mode tersebut bisa diberikan operasi or (|).

Selengkapnya tentang mode-mode tersebut, bacalah manual stat (section 2).

Menghapus file

Di bagian ini, kita akan membahas penghapusan file (2.c). System call yang dipergunakan adalah unlink().

Berikut ini adalah source code 2.c:

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

int main (int argc, char * argv[])
{
  int res;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
   exit (1);
  }

  res = unlink (argv[1]);

  if (res == 0)
  {
   fprintf (stdout, "File %s deleted
successfully\n", argv[1]);
  }
  else
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (-1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 2 2.c
```

Contoh output:

```
$ ./2 /bin/ls
Permission denied

$ ./2 notfound
No such file or directory

$ ./2 dir1
Is a directory

$ ./2 xxxx
File xxxx deleted successfully
```

Penjelasan source code:

- Pertama-tama, kita memeriksa jumlah argumen. File yang akan dihapus akan diberikan lewat argumen program (argv[1]).
- Ketika unlink() sukses dilakukan, maka nilai 0 akan dikembalikan. Ketika gagal, nilai -1 akan dikembalikan.

Mengetahui ukuran file

Di bagian ini, kita akan membahas bagaimana mendapatkan ukuran sebuah file (3.c). System call yang dipergunakan adalah fstat().

Berikut ini adalah source code 3.c:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

int main (int argc, char * argv[])
{
  int fd;
  int res;

  struct stat stat_buf;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
```

IKLAN

```
    exit (1);
  }

  fd = open (argv[1], O_RDONLY);

  res = fstat (fd, &stat_buf);

  if (res == 0)
  {
   fprintf (stdout, "Size of %s is
%ld byte\n", argv[1], stat_buf.
st_size);
  }
  else
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (-1);
  }

  close (fd);

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 3 3.c
```

Contoh output:

```
$ ./3 /bin/ls
Size of /bin/ls is 72468 byte
```

Penjelasan source code:

- Untuk mengetahui ukuran, kita dapatkan dari *member* st_size milik struct stat.
- Struktur data struct stat bisa dibaca di manual stat (section 2).

Memeriksa akses terhadap file

Sebelum kita membaca atau menulis suatu file, ada baiknya kalau kita periksa dahulu hak kita akan file tersebut. Di bagian ini, dengan mempergunakan system call access(), diperiksa hak kita akan suatu file (4.c).

Berikut ini adalah source code 4.c:

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

int main (int argc, char * argv[])
{
  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
    exit (1);
  }

  if (access (argv[1], F_OK) == 0)
  {
   fprintf (stdout, "%s exists\n",
argv[1]);
  }

  if (access (argv[1], R_OK) == 0)
  {
   fprintf (stdout, "%s is readable\
n", argv[1]);
  }

  if (access (argv[1], W_OK) == 0)
  {
   fprintf (stdout, "%s is
writeable\n", argv[1]);
  }

  if (access (argv[1], X_OK) == 0)
  {
   fprintf (stdout, "%s is
executable\n", argv[1]);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah:

```
$ gcc -o 4 4.c
```

Contoh output:

```
$ ./4 /bin/ls
/bin/ls exists
/bin/ls is readable
/bin/ls is executable

$ ./4 4
4 exists
4 is readable
4 is writeable
4 is executable
```

Penjelasan source code:

- Untuk informasi tentang mode yang ingin diuji, bacalah manual access (*section* 2).

Membaca link

Di Linux, kita mengenal file spesial *symbolic link*. Di bagian ini, kita akan membahas bagaimana membaca file yang ditunjuk oleh sebuah symlink (5.c). System call yang dipergunakan adalah readlink().

Berikut ini adalah source code 5.c:

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

int main (int argc, char * argv[])
{

  char ref_file[256];
  int len;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
   exit (1);
  }

  len = readlink (argv[1], ref_file,
sizeof (ref_file));

  if (len == -1)
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (-1);
  }
  else
  {
   ref_file[len] = '\0';
   fprintf (stdout, "%s is symlink
to %s\n", argv[1], ref_file);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 5 5.c
```

Contoh output:

```
$ ./5 5
Invalid argument

$ ./5 /bin/sh
/bin/sh is symlink to bash
```

Penjelasan source code:

- readlink() akan mengembalikan jumlah karakter yang disimpan di *buffer* apabila sukses dan -1 apabila gagal.
- Harap diperhatikan, readlink() tidak menambahkan *null* ke buffer.

Mengopi file

Di bagian ini, kita akan membahas proses pengopian file dengan system call read() dan write(), pengopian dengan blok

sebesar 4096 byte (6.c).

Berikut ini adalah source code 6.c:

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/stat.h>

int main (int argc, char * argv[])
{

  char buf[4096];
  int len;

  int f1;
  int f2;

  struct stat stat_buf;

  if (argc != 3)
  {
   fprintf (stderr, "usage %s <src_
file> <dst_file>\n", argv[0]);
   exit (1);
  }

  f1 = open (argv[1], O_RDONLY);
  if (f1 == -1)
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (2);
  }
  fstat (f1, &stat_buf);

  f2 = open (argv[2], O_WRONLY |
O_CREAT, stat_buf.st_mode);
  if (f2 == -1)
  {
   fprintf (stderr, "%s\n",
strerror(errno));
   exit (3);
  }

  while ( len = read (f1, buf,
sizeof (buf))  )
  {
   if (len != sizeof (buf))
     write (f2, buf, len);
   else
     write (f2, buf, sizeof (buf));
  }

  close (f1);
  close (f2);

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 6 6.c
```

Penjelasan source code:

- Apabila sukses, kita tidak menampilkan informasi apapun.
- Argumen 1 (argv[1]) adalah file sumber. Kita akan membuka dengan flag O_RDONLY.
- Argumen 2 (argv[2]) adalah file target. Kita akan membuka dengan flag O_WRONLY | O_CREAT. Harap diperhatikan, menggunakan fungsi fstat (lihat pembahasan tentang mengetahui ukuran file), kita akan mendapatkan mode file sumber, sehingga file target memiliki mode yang sama dengan file sumber.
- Logika pengopian:
  - Ulangi selama pembacaan dengan read() masih mengembalikan nilai non-zero (berarti masih ada yang harus dibaca).
  - Jika ukuran yang dikembalikan sama dengan panjang buffer, maka kita akan menulis sejumlah buffer. Namun, kalau tidak sama (misal pada akhir file di mana jumlah yang terbaca lebih kecil dari ukuran buffer), maka kita akan menulis sejumlah yang terbaca.
- Bisa dilihat bahwa program kita bisa berjalan cukup cepat, apabila dibandingkan dengan cp (pada percobaan pengopian dua file berukuran 100 MB):

```
$ dd if=/dev/zero of=/tmp/test_cp
bs=1M count=100

$ dd if=/dev/zero of=/tmp/test_6
bs=1M count=100
```

Percobaan dengan cp:

```
$ time cp /tmp/test_cp test_cp

real    0m1.612s
user    0m0.044s
sys     0m1.504s

$ time ./6 /tmp/test_6 test_6

real    0m1.699s
user    0m0.024s
sys     0m1.596s
```

- Ingin lebih cepat lagi?
  - Perbesar buffer. Contoh dengan buffer 10 kali lipat (penulis gunakan 6b):

```
$ time ./6b /tmp/test_6 test_6

real    0m0.771s
user    0m0.008s
sys     0m0.696s
```

  - Gunakan system call sendfile(), yang bekerja di kernel. Mantap. Selengkapnya, bacalah manual sendfile (section 2).

```
ssize_t sendfile(int out_fd,
int in_fd, off_t *offset,
size_t count);
```

Memindahkan/mengubah nama

Di bagian ini, kita akan membahas bagaimana memindahkan/mengubah nama file, mempergunakan system call rename (7.c).

Berikut ini adalah source code 7.c:

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

int main (int argc, char * argv[])
{

  int res;

  if (argc != 3)
  {
   fprintf (stderr, "usage %s <old_
path> <new_path>\n", argv[0]);
   exit (1);
  }

  res = rename (argv[1], argv[2]);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah:

```
$ gcc -o 7 7.c
```

Penjelasan source code:

- Apabila sukses, kita tidak menampilkan

pesan apapun.

- System call *rename* sangat mudah digunakan, dengan menerima dua parameter berupa old_path dan new_path.

Mengubah mode file

Di bagian ini, kita akan mengubah mode file dengan system call chmod(). Secara sederhana, kita akan mengubah mode ke 0, yang artinya tidak membuat user, group dan *other* tidak memiliki hak apapun.

Berikut ini adalah source code 8.c:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>

int main (int argc, char * argv[])
{

  int res;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
   exit (1);
  }

  res = chmod (argv[1], 0);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah:

```
$ gcc -o 8 8.c
```

Penjelasan source code:

- apabila sukses, chmod akan mengembalikan 0.
- Untuk mode selengkapnya, bacalah manual chmod (section 2).

Mengubah kepemilikan file

Di bagian ini, kita akan mengubah kepemilikan file menggunakan system call chown. Secara sederhana, kita akan mengubah kepemilikan menjadi user root dan group root. Untuk itu, hak user root diperlukan.

Berikut ini adalah source code 9.c:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char * argv[])
{

  int res;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <file>\
n", argv[0]);
   exit (1);
  }

  res = chown (argv[1], 0, 0);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 9 9.c
```

Contoh output di mana hak tidak dimiliki:

```
$ ./9 9
Operation not permitted
```

Penjelasan source code:

- parameter kedua chown() dalam contoh ini adalah uid user. Parameter ketiga adalah gid group.
- Ketika sukses, 0 akan dikembalikan.
- Untuk tidak melakukan penggantian user atau group, berikanlah nilai -1 sebagai parameter.

Masuk ke direktori

Di bagian ini, kita akan membahas contoh untuk masuk ke direktori tertentu menggunakan system call chdir(). Berikut ini adalah source code 10.c:

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

int main (int argc, char * argv[])
{

  int res;
  char buf[256];

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <dir>\
n", argv[0]);
   exit (1);
  }

  res = chdir (argv[1]);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }
  else
  {
   getcwd (buf, sizeof (buf));
   fprintf (stdout, "CWD: %s\n",
buf);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan perintah berikut:

```
$ gcc -o 10 10.c
```

Contoh output:

```
$ ./10  /dir-yang-tidak-ada
No such file or directory

$ ./10  /
CWD: /
```

Penjelasan source code:

- Apabila sukses, chdir() akan mengembalikan 0.
- Pada percobaan kita, apabila sukses, kita akan menampilkan direktori aktif, yang bisa didapatkan dengan getcwd().
- Bacalah manual chdir (section 2), dan getcwd (section 3).

Membuat direktori

Di bagian ini, kita akan membahas pembuatan direktori dengan system call mkdir(). Berikut ini adalah source code 11.c:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>

int main (int argc, char * argv[])
{

  int res;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <dir>\
n", argv[0]);
   exit (1);
  }

  res = mkdir (argv[1], S_IRWXU);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan:

```
$ gcc -o 11 11.c
```

Penjelasan source code:

- Apabila sukses, mkdir() akan mengembalikan 0.
- Parameter kedua mkdir() di contoh ini adalah mode. Bacalah manual stat (section 2) untuk informasi selengkapnya.

Menghapus direktori

Di bagian ini, kita akan membahas penghapusan direktori dengan system call rmdir(). Direktori yang ingin dihapus sudah harus kosong terlebih dahulu. Berikut ini adalah source code 12.c:

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

int main (int argc, char * argv[])
{

  int res;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <dir>\
n", argv[0]);
   exit (1);
  }

  res = rmdir (argv[1]);

  if (res != 0)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (1);
  }

  return 0;

}
```

Lakukanlah kompilasi dengan

```
$ gcc -o 12 12.c
```

Menampilkan isi direktori

Di bagian terakhir ini, kita akan mendapatkan isi direktori dengan fungsi standar opendir(), readdir() dan closedir(). Berikut ini adalah source code 13.c:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <dirent.h>

int main (int argc, char * argv[])
{

  DIR *d;

  struct dirent * direntry;

  if (argc != 2)
  {
   fprintf (stderr, "usage %s <dir>\
n", argv[0]);
   exit (1);
  }

  d = opendir (argv[1]);

  if (d == NULL)
  {
   fprintf (stderr, "%s\n", strerror
(errno));
   exit (2);
  }

  direntry = readdir (d);

  while (direntry  != NULL)
  {

   fprintf (stdout, "%s\n", direntry
-> d_name);
   direntry = readdir (d);
  }

  closedir(d);

  return 0;

}
```

Lakukanlah kompilasi dengan:

```
$ gcc -o 13 13.c
```

Contoh output:

```
$ ./13 /
.
..
dev
tmp
var
bin
etc
lib
mnt
sys
usr
boot
home
proc
sbin
root
```

Untuk informasi selengkapnya, bacalah manual section 3, yakni opendir, readdir dan closedir. Pemrograman direktori dan file, cukup banyak digunakan pada kebanyakan aplikasi yang ada di Linux. Silakan dikembangkan lebih lanjut, untuk disesuaikan dengan kebutuhan aplikasi Anda.

Sampai di sini dulu pembahasan kita. Sampai ketemu lagi di pembahasan system call Linux lainnya. Selamat mencoba.

Noprianto [noprianto@infolinux.co.id]

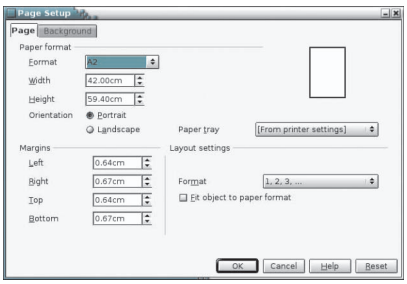
Membuat Kalender Dengan Open-Office.org

Walaupun OpenOffice.org Draw memiliki tampilan yang cukup sederhana dibandingkan dengan *software* seperti CorelDraw, Illustrator atau Freehand, kemampuan OpenOffice.org Draw tidak perlu diragukan lagi dalam membuat objek-objek berjenis vektor. Satu kemampuannya adalah untuk membuat kalender secara cepat dan mudah.

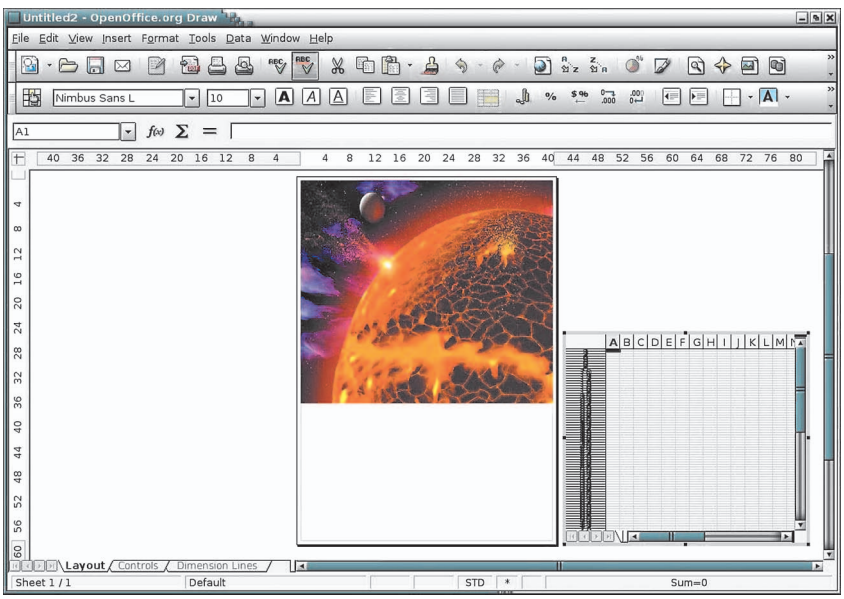
Mungkin banyak sebagian orang menganggap bahwa Linux terbelakang dalam soal desain grafis, ternyata itu semuanya tidak benar. Selain OpenOffice.org Draw, masih terdapat pula aplikasi desain grafis lainnya seperti Sodipodi (setaraf dengan CorelDraw), Scribus (sama dengan Pagemaker), GIMP (sama dengan Photoshop). Untuk aplikasi rancang bangun pun, Linux telah memiliki Qcad yang hampir sama dengan AutoCad, namun Qcad hanya belum memiliki fasilitas dalam hal membuat objek 3D.

Berikut akan dijelaskan bagaimana cara membuat kalender dengan menggunakan OpenOffice.org Draw. Proses kerjanya adalah sebagai berikut:

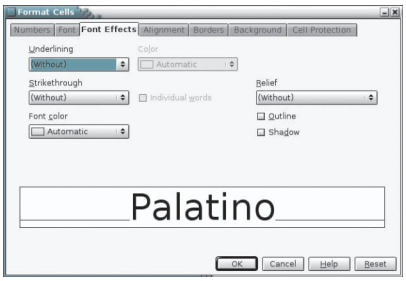
1. Aturlah ukuran kertas yang dikehendaki, misalnya kita akan menggunakan ukuran A2 (42 cm x 59 cm). Caranya yaitu dengan mengklik *Format* > *Page*... maka akan tampil kotak dialog Page Setup, seperti yang terlihat pada gambar 1.
2. Pilih A2 pada bagian *Format* dan *Orientation*-nya *Potrait*. Kemudian sisipkanlah *image* yang akan digunakan dalam menghiasi suatu kalender, dengan cara mengklik *Insert* > *Picture* > *From File*. Lalu aturlah tata letaknya sehingga terlihat menarik.
3. Berikutnya, setelah menyisipkan image, yaitu membuat tanggal, bulan dan tahun pada kalender tersebut. Caranya yaitu dengan mengklik *Insert* > *Spreadsheet*, maka akan terlihat lembar kerja Excell, seperti yang terlihat pada gambar 2.
4. Aturlah ukuran dan jenis *font* yang dikehendaki. Di sini, penulis menggunakan jenis huruf Palatino dengan ukuran 44. Sedangkan untuk ukuran kolom, yaitu 7.5 cm, caranya dengan mengklik *Format* > *Column* > *Width*... dengan ketentuan *spreadsheet* atau lembar kerja masih aktif. Buatlah nama-nama hari dalam seminggu tersebut pada baris ketiga.
5. Umumnya, warna hari-hari dalam seminggu pada kalender itu tidak sama. Misalnya kita akan mengubah warna hari Jum'at dengan warna hijau dan hari Minggu dengan warna merah. Caranya yaitu dengan mengklik *Format* > *Cells*... maka akan tampil kotak dialog Format Cells, seperti yang terlihat pada gambar 3.
   Kemudian pilih tab *Font Effects*, pilihlah warna yang dikehendaki. Namun



Gambar 1. Kotak dialog Page Setup.



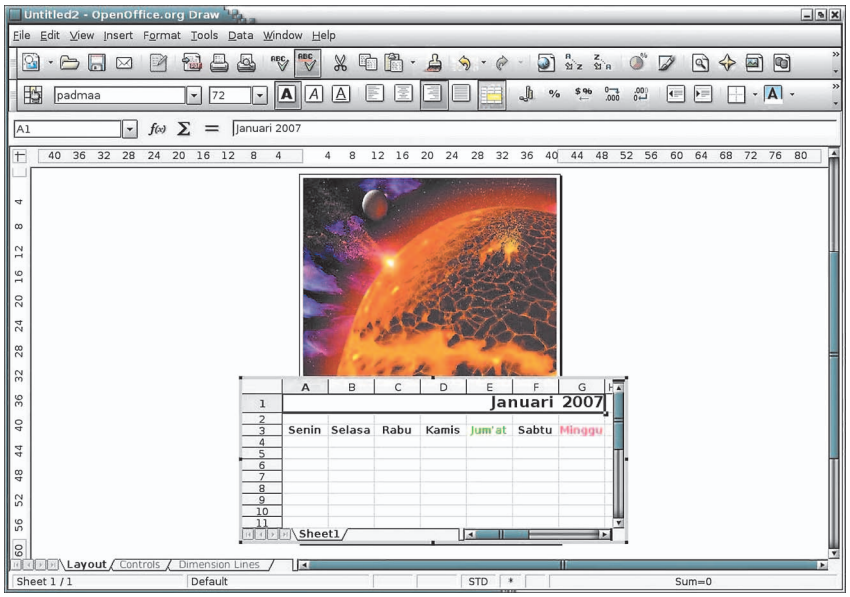
Gambar 2. Spreadsheet atau lembar kerja.



Gambar 3. Kotak dialog Format Cells.

dengan catatan teks yang akan diubah warnanya harus dalam kondisi terblok atau terpilih.

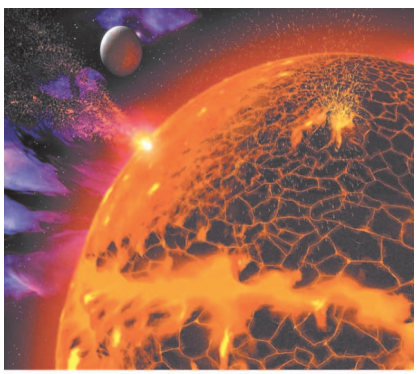
6. Berikutnya, pada baris pertama sisipi teks "Januari 2007" dengan ketentuan ubah ukuran dan jenis font-nya. Di sini, penulis menggunakan jenis font Padmaa dengan ukuran 72. Setelah itu tebalkan teks tersebut. Setelah menebalkan teks tersebut, maka langkah berikutnya yaitu mengatur letak teks tersebut di bagian sebelah kanan, yaitu tepat di hari Minggu. Caranya dengan memblok sel A1 sampai G1, lalu klik *Format* > *Merge Cells* dan *tool Align Right* atau klik *Format* > *Cells,* dan pilih tab *Alignment* pada kotak dialog Format Cells... Hasilnya akan terlihat seperti gambar 4.
7. Bloklah sel A3 sampai G3 yang berisikan teks Senin sampai Minggu, lalu ubah warna dasarnya menjadi kuning. Caranya dengan mengklik tool *Background color* atau klik Format > Cells... dan pilih tab Background.
8. Selanjutnya, bloklah sel A4 sampai G8, dan kemudian klik tool *Center Align* dan *Bold*. Berikutnya, sisipkanlah angka 1 sampai 31. Ubahlah warna angka-angka yang berada satu kolom dengan hari Minggu dan tanggal 21 dengan warna merah.
9. Kini saatnya kita memberikan *border* untuk nama-nama hari dan tanggal-tanggal dalam sebulan tersebut, agar terlihat menarik. Untuk memberikan border pada nama-nama hari dalam seminggu, caranya, dengan memblok sel A3 sampai G3, kemudian klik *pulldown* tool Borders, dan pilih tipe border yang meliputi garis atas-bawah dan kiri-kanan, atau dapat juga dengan cara mengklik Format > Cells..., kemudian pilih tab *Borders* dan klik *Set Outer Border Only* pada *Line arrangement*.



Gambar 4. Tampilan spreadsheet setelah proses pembuatan penanggalan.

10. Setelah memberikan border pada nama-nama hari dalam seminggu, maka langkah berikut memberikan border pada tanggal-tanggal dalam sebulan. Caranya, bloklah sel A4 sampai G7, setelah itu klik pulldown tool Borders, dan pilih tipe border yang meliputi keseluruhan sel, yaitu atas-bawah, kiri-kanan, tengah-horizontal dan tengah-vertikal.
11. Lakukan cara yang sama seperti pada no. 10, untuk sel A8 sampai C8, maka hasilnya akan terlihat seperti gambar 5.
12. Kemudian langkah terakhir, yaitu memberikan keterangan untuk tanggal-tanggal yang dianggap libur nasional atau perayaan keagamaan. Pada baris ke sembilan, perkecil tingginya menjadi 1 cm, dan kemudian pada sel A10, sisipkanlah teks "1 Januari: Tahun Baru Masehi", dan ubah ukuran fontnya menjadi 36. Selanjutnya, masih pada sel yang sama, bloklah sel A10 sampai G10, dan kemudian lakukan *merge* sel. Lalu, klik tool Align Right.
13. Akhirnya, sisipkanlah teks "20 Januari: Tahun Baru Hijriah (1 Muharam 1428 H)" pada sel A11. Lakukan cara yang sama seperti pada nomor 12, maka hasilnya akan terlihat seperti gambar 6.

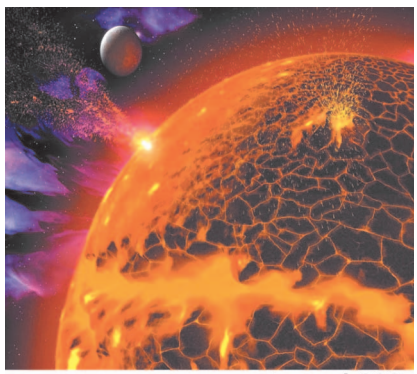
Mohamad Sukarno [karno180372@gmail.com]



Januari 2007

Senin	Selasa	Rabu	Kamis	Jum'at	Sabtu	Minggu
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	23	25	26	27	28
29	30	31				

Gambar 5. Tampilan kalender sementara.



Januari 2007

Senin	Selasa	Rabu	Kamis	Jum'at	Sabtu	Minggu
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	23	25	26	27	28
29	30	31				

1 Januari: Tahun Baru Masehi
20 Januari: Tahun Baru Hijriah (1 Muharram 1428 H)

Gambar 6. Tampilan kalender tahap akhir.

Konfigurasi NFS Server Menggunakan CentOS 5.0

NFS merupakan aplikasi *server* yang dapat kita manfaatkan untuk saling berbagi file. Dengan NFS, komputer Anda dapat melakukan *mounting* komputer orang lain atau server, melalui proses *remote* sehingga memudahkan proses berbagi data antarkomputer yang berbasis Unix.

Pada platform Linux, terdapat dua protokol yang cukup populer untuk dapat saling berbagi file. Dua protokol yang dimaksud adalah SMB dan NFS. Keduanya merupakan protokol yang sudah sangat mapan, dan masih terus dikembangkan. Protokol SMB sangat populer digunakan, karena mayoritas pengguna komputer di dunia saat ini masih banyak yang menggunakan Windows. Sedangkan protokol NFS, sangat populer digunakan pada platform berbasis Unix/Linux. Meski demikian, implementasi SMB sudah dapat diimplementasikan dengan baik di Linux melalui project SAMBA.

Jika mayoritas sistem operasi yang digunakan para pengguna komputer dalam jaringan berbasiskan Linux, penggunaan protokol NFS akan lebih tepat digunakan daripada menggunakan protokol SMB. Namun jika OS Client dalam suatu jaringan mayoritas lebih banyak yang menggunakan sistem operasi Windows, maka penggunaan SAMBA untuk menjalankan protokol SMB akan lebih tepat untuk diterapkan.

Pada tutorial kali ini, *InfoLINUX* akan menjelaskan proses konfigurasi NFS server, berikut proses *setting* NFS di komputer *client*. Sebagai bahan percobaan untuk pembuatan artikel ini, *InfoLINUX* menggunakan distro CentOS 5.0 yang telah disertakan sebagai bonus DVD *InfoLINUX* 06/2007.

Persiapan

1. Dimisalkan Anda sudah dapat menginstalasikan distro CentOS 5.0 dengan baik.
2. Sebagai bahan percobaan, *InfoLINUX* mengasumsikan PC yang digunakan dalam artikel adalah sebagai berikut.
   - Server memiliki IP:

     ```
     192.168.2.1
     ```
   - *Workstation*/PC Client, masing-masing memiliki IP:

     ```
     192.168.2.3
     192.168.2.4
     ```

Setelah PC server dan client telah dikonfigurasi sesuai dengan petunjuk IP di atas. Berikutnya kita akan mengonfigurasi NFS Server.

Konfigurasi NFS Server

Kita akan menjadikan PC dengan IP 192.168.2.1 sebagai NFS server. Untuk itu, semua perintah di bawah ini akan kita jalankan pada PC tersebut.

1. Untuk mempermudah instalasi paket NFS, Anda dapat menggunakan aplikasi Yum. Untuk mengonfigurasikan Yum, lakukan langkah sebagai berikut:
   - Masukan DVD CentOS 5.0 ke dalam DVD-ROM drive.
   - Jalankan aplikasi Gnome Terminal, kemudian lihat di mana DVD CentOS 5.0 ter-*mount* dengan menggunakan perintah df -h. Dalam contoh ini, DVD CentOS 5.0 ter-mount sebagai /media/CentOS-5-IL062007.
   - Edit file /etc/yum.conf sehingga terlihat sebagai berikut.

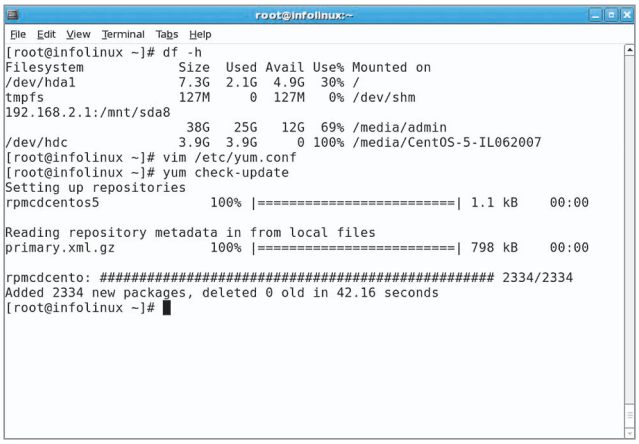
     ```
     .....
     gpgcheck=0
     .....
     # PUT YOUR REPOS HERE OR IN
     separate files named file.repo
     # in /etc/yum.repos.d
     [rpmcdcentos5]
     name=rpmcdcentos5
     baseurl=file:///media/CentOS-5-
     IL062007
     enabled=1
     ```
   - Terakhir, update database paket Yum dengan menggunakan perintah berikut:

     ```
     # yum check-update
     ```
2. Setelah Yum terkonfigurasi dengan baik, berikutnya Anda dapat menginstalasikan paket NFS dengan menggunakan perintah berikut:

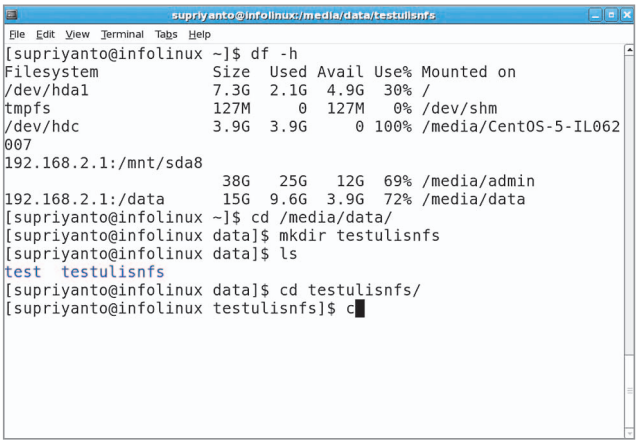
   ```
   # yum install nfs-utils nfs-
   utils-lib portmap system-config-
   nfs
   ```
3. Berikutnya, kita akan mengonfigurasi file /etc/exports, dengan parameter yang berisi folder yang ingin di-*share*, PC dengan IP berapa yang diperbolehkan untuk mengakses folder yang di-share, dan hak *permission*-nya.

   Sebagai contoh, di bawah ini kita akan melakukan share folder/data kepada PC dengan IP 192.168.2.3 dan 192.168.2.4, yang diberikan akses *read/write*, dan akses *root* dalam server.

   ```
   /data   192.168.2.3(rw,no_root_
   squash,sync) 192.168.2.4(rw,no_
   root_squash,sync)
   ```



Setup Yum di PC NFS Server.



Tes tulis di NFS share.

Penjelasan:

- Perintah di atas diketik dalam satu baris perintah.
- Parameter pertama yang terdapat pada baris di atas menerangkan direktori yang akan di-share.
- Parameter kedua menerangkan IP berapa saja yang diperbolehkan untuk mengakses direktori tersebut, dan bagian dalam *bracket* mengizinkan akses read/write, dan akses root dalam server (*entry no_root_squash* bersifat *optional*).

  Jika Anda mengalami kesulitan karena tidak terbiasa dengan mengetikan perintah di Terminal, Anda dapat menggunakan aplikasi NFS Server Configuration yang berbasis GUI untuk mengonfigurasi file export. Aplikasi ini dapat Anda temukan dari menu *System -> Administration -> Server Settings -> NFS*.

4. Lanjutkan dengan mengedit file /etc/hosts.allow.

```
# vim /etc/hosts.allow
```

Paste baris berikut pada file /etc/hosts.allow. Dalam contoh ini, hanya PC dengan IP 192.168.2.3 dan 192.168.2.4, yang diperbolehkan untuk mengakses server.

```
Portmap: 192.168.2.3, 192.168.2.4
lockd: 192.168.2.3, 192.168.2.4
mountd: 192.168.2.3, 192.168.2.4
rquotad: 192.168.2.3, 192.168.2.4
statd: 192.168.2.3, 192.168.2.4
```

Simpan hasil perubahan yang telah dilakukan.

5. Lanjutkan dengan mengedit file /etc/hosts.deny.

```
# vim /etc/hosts.deny
```

Paste baris berikut pada file /etc/hosts.deny.

```
portmap: ALL
lockd: ALL
mountd: ALL
rquotad: ALL
statd: ALL
```

Baris di atas menjaga semua *hosts* yang tidak terdaftar dalam daftar file /etc/hosts.allow, untuk dapat mengakses server.

6. Langkah berikutnya adalah mengekspor direktori yang ingin di-share, dan menjalankan *service* NFS. Untuk memastikan agar service NFS berjalan setiap kali *reboot*, jalankan perintah berikut dari Terminal.

```
# chkconfig nfs on
```

Kemudian export direktori dari /etc/exports, dengan menggunakan perintah berikut:

```
# exportfs -ra
```

Reboot dan service NFS akan berjalan secara otomatis.

Setting NFS Klien

Tahapan selanjutnya setelah menginstalasi NFS Server adalah mengonfigurasikan NFS klien. Untuk itu, pindah ke salah satu PC yang akan digunakan sebagai NFS klien, kemudian lakukan langkah sebagai berikut.

1. Langkah menginstalasi paket NFS yang dibutuhkan NFS klien kurang lebih sama dengan yang sudah kita lakukan pada NFS server. Untuk itu, pada PC yang menjalankan NFS klien, silakan jalankan perintah berikut untuk menginstalasi paket yang dibutuhkan oleh NFS klien.

```
# yum install nfs-utils nfs-
utils-lib portmap system-config-
nfs
```

2. Buat sebuah *mounpoint* untuk mengakses NFS share. Dalam contoh ini, kita akan membuat sebuah direktori bernama /media/admin, yang akan digunakan sebagai mountpoint NFS share.

```
# mkdir -p /media/admin
```

Ubah juga permission mount point direktori tersebut, sehingga Anda tetap dapat mengaksesnya sebagai user biasa.

```
# chmod 777 /media/admin
```

3. Tambahkan sebuah *entry* sebagai berikut pada file /etc/fstab.

```
# vim /etc/fstab
.....
.....
192.168.2.1:/data /media/admin
nfs noauto,rw,user 0 0
```

Lakukan proses mount direktori yang terdapat pada NFS server.

```
# mount -a
```

4. Anda juga dapat memastikan apakah service NFS server sedang berjalan, atau tidak dengan menggunakan perintah berikut:

```
# rpcinfo -p 192.168.2.1
```

Folder share yang terdapat pada NFS server, juga dapat Anda mount dengan menggunakan perintah berikut:

```
# mount /media/admin
```

Sekarang Anda sudah dapat mengakses direktori /data yang terdapat pada folder share NFS server pada direktori mount point /media/admin. Tes dengan membaca atau menuliskan suatu file pada direktori mount point tersebut. Jika berhasil, berarti Anda sudah berhasil melakukan konfigurasi NFS server ini dengan benar.

Supriyanto [supriyanto@infolinux.co.id]

Backup dan Restore Database MySQL Menggunakan Zmanda

Tidak peduli apakah Anda membuat *blog* berisi catatan harian, atau membuat aplikasi *corporate* besar yang menyimpan jutaan data *customer*, melakukan proses *backup* data di suatu *database* merupakan suatu keharusan. Dengan ini, proses *recovery* akan mudah dilakukan, jika suatu saat terjadi kerusakan *hardware*.

MySQL dikenal sebagai database *open source* yang banyak digunakan untuk berbagai kebutuhan, mulai dari aplikasi sistem, *website*, hingga untuk menyimpan *log server*. Tak heran jika solusi backup yang dibuat pihak *third party software* untuk database MySQL juga cukup beragam.

Secara umum, data yang terdapat dalam database MySQL dapat dengan mudah kita backup, dengan cara melakukan *dump* isi tabel atau database ke dalam sebuah file teks. Meski perintah ini dapat kita lakukan dengan mudah, namun kita harus selalu melakukan hal ini, setiap saat ingin melakukan proses backup. Meskipun hal ini dapat ditangani dengan menggunakan *scheduler* seperti *cron*, untuk menjalankan proses backup secara otomatis dalam kurun waktu tertentu.

Cara lainnya, kita dapat menggunakan sejumlah aplikasi yang telah dibuat oleh pihak ketiga, untuk membantu kita dalam urusan backup data yang terdapat di dalam database MySQL. Salah satunya adalah dengan menggunakan ZMR (*Zmanda Recovery Manager*). Aplikasi ini masih termasuk dalam *project* Zmanda, yang dikenal dengan rangkaian aplikasi backup yang bersifat open source.

ZMR-MySQL mengizinkan *administrator* database agar secara otomatis menjalankan logical, atau *raw* backup *process* ke lokal atau *remote disk*, dan mengizinkan untuk melakukan recovery database dengan mudah kapan saja.

Dalam kesempatan ini, *InfoLINUX* akan membahas cara penggunaan ZMR untuk mem-backup suatu data yang terdapat dalam database MySQL.

Pra Instalasi

Sebelum menginstalasi ZRM, kita harus menginstalasi dan mengonfigurasi beberapa hal sebagai berikut:

1. MySQL Server.
   Pastikan database MySQL server telah terinstal dengan baik. Selain itu, Anda juga perlu menambahkan beberapa parameter file konfigurasi /etc/my.cnf sebagai berikut:
   - log-bin
     Tambahkan baris *log-bin*, pada [mysqld] *section*.

```
log-bin
```

   - max_binlog_size
     Sebagai tambahan, untuk database yang berukuran besar, Anda dapat menambahkan parameter max_binlog_size = <ukuran max size> di bawah parameter log-bin. Hasil konfigurasi akhir pada file /etc/my.cnf akan terlihat sebagai berikut:

```
[mysqld]
.............................
log-bin
max_binlog_size=2G
```

   Setelah melakukan konfigurasi, lakukan proses *restart service* database MySQL.

```
# /etc/init.d/mysqld restart
```

2. Perl module.
   Paket ZRM membutuhkan beberapa *module* Perl, seperti perl-DBI dan perl-XML-parser.

Instalasi ZRM

Paket ZRM tersedia dalam bentuk *binary* dan source. Karena dalam percobaan ini *InfoLINUX* menggunakan Fedora Core 6, maka dalam contoh ini proses instalasi menggunakan paket binary dalam bentuk RPM.

- *Download* file ZRM pada situs *http://mysqlbackup.zmanda.com*, atau Anda dapat menemukan paket ZRM pada DVD *InfoLINUX* edisi ini dalam folder "rubrik/tutorial".
- Instalasikan paket ZRM dengan menggunakan perintah berikut:

```
# rpm -ivh MySQL-zrm-1.1.5-
1.noarch.rpm  MySQL-zrm-socket-
server-1.1.5-1.noarch.rpm
```

Setelah terinstal, paket zrm dapat ditemukan pada direktori /usr/bin. Sedangkan file konfigurasi ZRM dapat ditemukan pada direktori /etc/mysql-zrm.

Konfigurasi ZRM

Berikutnya kita akan mulai melakukan konfigurasi ZRM. Untuk itu, lakukan langkah sebagai berikut:

- Buat sebuah direktori yang bernama dailyrun, dan *copy* contoh file kon-

figurasi mysql-zrm.conf ke direktori tersebut.

```
# cd /etc/mysql-zrm
# mkdir dailyrun
# cp /etc/mysql-zrm/mysql-zrm.
conf/etc/mysql-zrm/dailyrun
```

- Buka file /etc/mysql-zrm/mysql-zrm.conf, dan edit beberapa parameter pada file konfigurasi tersebut. Caranya, hapus tanda "#" yang menandakan komentar pada file mysql-zrm.conf.

```
Backup-level=0
backup-mode=logical
retention-policy=10D
compress=1
# databases="database yang ingin
dibackup"

# Sebagai contoh, database yang
ingin dibackup adalah zabbix
dan webo. Username dan password
database tersebut adalah "user1"
dan "test123".
databases=zabbix webo
user="user1"
password="test123"
verbose=1
mailto="your@email.addr"
```

Simpan dan keluar dari file konfigurasi yang telah diedit.

Menjalankan proses backup

Untuk memulai proses backup dengan ZRM, lakukan sejumlah langkah sebagai berikut:

- Ketikkan perintah berikut untuk memulai proses backup harian untuk database, yang telah Anda definisikan pada file /etc/mysql-zrm/dailyrun/mysql-zrm.conf.

```
# mysql-zrm-scheduler --now --
backup-set dailyrun
```

- Setelah mengetikkan perintah di atas, akan tampil informasi dari hasil perintah tersebut. Baca baik-baik hasilnya apakah terdapat pesan *error* atau *warning* string yang ditampilkan.

Penjadwalan backup

ZRM dapat dijadwalkan agar secara otomatis dapat menjalankan proses backup, pada waktu yang ditentukan. Sebagai contoh, jika ingin melakukan proses backup setiap hari pada pukul 04:30 am, Anda dapat menjalankan perintah berikut:

```
# mysql-zrm-scheduler --add --
interval daily --start 04:30 --
backup-level 0 --backup-set
dailyrun
```

Untuk mengecek kalau jadwal *schedule* perintah di atas sudah benar, jalankan perintah berikut:

```
# mysql-zrm-scheduler --query
```

Hasil perintah di atas akan menampilkan *path logging schedule time* yang menyerupai cron.

Untuk menampilkan ringkasan backup yang telah dikerjakan, jalankan perintah berikut:

```
# mysql-zrm-reporter --where
backup-set=dailyrun --show backup-
status-info
```

Hasil perintah di atas akan mencetak laporan backup_set, backup_date, backup_level, dan backup_status.

Restore Database

Untuk melakukan proses *restore* database yang telah didefinisikan pada file konfigurasi ZRM, pertama, tentukan dahulu backup directory mana yang akan di-restore. Untuk mendapatkan informasi mengenai hal ini, Anda dapat menjalankan perintah berikut:

```
# mysql-zrm-reporter --show
restore-info --where backup-
set=dailyrun
```

Perintah di atas akan menghasilkan daftar backup directory yang ada pada ZRM. Pilih backup directory yang ingin di-restore. Sebagai contoh di sini nama backup directory yang akan di-restore adalah /var/lib/mysql-zrm/dailyrun/20070606043002. Untuk memulai proses restore, jalankan perintah berikut:

```
# mysql-zrm --action restore -
-backup-set dailyrun --source-
directory /var/lib/mysql-zrm/
dailyrun/20070606043002/
```

Sekarang Anda dapat melakukan proses backup dan restore database MySQL dengan mudah. Dengan menggunakan ZRM, risiko hilangnya data karena kerusakan hardware dapat lebih diminimalisasi. Untuk mendapatkan keterangan lebih lengkap mengenai ZRM, silakan membaca manual Administration ZRM yang terdapat di situsnya.

Supriyanto [supriyanto@infolinux.co.id]