

Seluk-Beluk INSTALASI

Noprianto dan Salman Agus Supriadi

Dunia berubah, kebutuhan manusia pun bertambah. Beberapa saat yang lalu, ketika kebutuhan kita akan komputasi masih belum terlalu menuntut, program-program yang telah terinstalasi di komputer pun terasa sudah sangat mencukupi. Mau mengetik, ada teks editor. Mau mendengarkan lagu, ada mp3 player yang siap membunyikan speaker kita. Hampir semua kebutuhan komputasi kita telah tersedia dalam sebuah CD-ROM distro.



Tapi sekarang, banyak hal sudah tidak sederhana lagi. Sementara distro masih menyediakan versi 1 dari program A, *upstream developer* dari program A telah merilis versi 2 yang jauh lebih merangsang, jauh lebih menggiurkan. Sementara itu, pemutar mp3 yang jauh lebih indah, jauh lebih *powerful* baru saja dirilis. Air liur pun menetes perlahan-lahan. Kita butuh sentuhan baru, kita butuh *upgrade*, kita perlu instalasi program baru.

Sekilas kedengarannya sederhana. Melakukan instalasi *software* sederhana bisa diartikan memasang software di sistem operasi kita. Akan tetapi, keterbukaan sistem operasi GNU/Linux membuat instalasi itu menjadi tidak semudah yang dibayangkan. Masalah-masalah bermunculan dan susah untuk dihindari. Mulai para sebagian jenius penulis software yang hanya mementingkan fungsi program dan menganggap instalasi adalah urusan sepele—*compile* sendiri dan semua urusan beres (kalau bermasalah,

pecahkan sendiri). Kemudian berpindah ke kebutuhan spesifik dan egois sistem operasi. Kemudian ada pula ketergantungan dengan pustaka-pustaka lain. Sampai kepada masalah upgrade ke versi yang lebih baru untuk mendapatkan hasil yang lebih *yahud*. Cukup kompleks, bukan?

Seseorang yang praktis bisa saja mengambil *source* dari *upstream developer*, membaca mati-matian semua manual yang tersedia ditambah mencari lagi ke forum dan milis, dan kemudian menginstalasi semua bagian dari software tersebut ke dalam satu lokasi tersendiri. *Path* pun bertambah. *Link* file ke manual pun bertambah. Keluar masuk direktori pun bertambah. Sekilas kelihatan sederhana karena kalau tidak senang dengan software tersebut, lokasi tersebut cukup dihapus saja. Adapula yang melakukan kompilasi dan instalasi sesuai *setting-an default*, yang akan berujung kepada sistem yang tidak stabil dan susah di-upgrade.

Sadar akan permasalahan tersebut,

pembuat distro mengembangkan *packet management* sendiri-sendiri (sebagian menggunakan milik distro lain). Tujuannya sama, yaitu membuat instalasi dan *tetek bengek*-nya menjadi mudah dan sederhana. Tidak perlu lagi melakukan kompilasi dari source, cukup jalankan program tertentu, berikan opsi yang bersesuaian dan software baru pun duduk dengan manis di file sistem kita. Software lama yang tidak dibutuhkan lagi pun dapat digusur dengan mudah.

Alangkah manisnya kalau instalasi semua software dapat diperlakukan dengan cara seragam. Sayangnya tidak, karena beberapa software menerapkan cara instalasi yang cukup melenceng dari software kebanyakan. Arrggggggghhh...

Untuk itu, di menu utama kali ini, kami membahas tuntas problem seputar instalasi dan packet management distro-distro agar instalasi software dapat dilakukan semudah menggerakkan mouse dan menekan beberapa tombol keyboard.

SERBA-SERBI RPM

Menginstalasi paket program bisa dengan sekali perintah. Demikian pula menghapusnya. Masalah baru muncul jika ada ketergantungan paket yang satu dengan yang lain. Bagaimana mengatasinya?

Manajemen paket *software* paling sering ditanyakan oleh para pengguna Linux pemula atau *newbie*, yang selalu membandingkan dengan kemudahan manajemen paket *software* di sistem operasi Windows. Salah satu manajemen paket yang cukup mudah digunakan adalah RPM atau RedHat Package Manager. *Software* ini hanya salah satu dari sekian banyak pilihan yang ada dalam distribusi Linux. Pilihan yang lain misalnya *pkgtool* milik distribusi Slackware.

Alasan memilih RPM

Sesuai dengan namanya, RPM atau Manajer Paket RedHat pertama dikeluarkan oleh distribusi Linux RedHat. Dari sekian banyak sistem manajemen paket yang ada, yang paling ampuh menurut penulis adalah RPM ini. Tidak heran jika distribusi-distribusi Linux selain RedHat juga menyertakannya, bahkan RPM dijadikan sebagai sistem manajemen *software* utama. Lihat saja distribusi SuSe dan Mandrake. Slackware yang sudah mempunyai sistem manajemen *software* sendiri juga masih menyertakan paket RPM.

File paket *software* yang dibuat dengan RPM bisa dikategorikan menjadi dua, yaitu paket dalam bentuk biner (*binary*) dan paket dalam bentuk kode sumber (*source*). Kalau di dalam paket biner berisi file-file yang siap diletakkan di posisi masing-masing, di dalam paket RPM kode sumber hanya berisi paket kode sumber aslinya—dalam bentuk *tarball*—file-file *patch* jika ada dan file *spec* yang bisa digunakan untuk membuat paket RPM biner. Informasi mengenai file spesifikasi atau file *spec* bisa Anda baca pada artikel “Membuat Paket RPM.”

Nama file paket RPM biner biasanya mempunyai format sebagai berikut.

`nama_software-versi-versi_rpm.ars.rpm`

Ars menunjukkan jenis prosesor, misalnya i386. Untuk paket rpm dalam bentuk *source*, format namanya sebagai berikut.

`nama_software-versi-versi_rpm.src.rpm`

Contoh-contoh file RPM biner dan *source*:

`dhcp-3.0.1rc9-1.i386.rpm`

`dhcp-3.0.1rc9-1.src.rpm`

`pdmnu-1.2.77-1.i386.rpm`

`pdmnu-1.2.77-1.src.rpm`

Bisa dilihat pada contoh, paket *software* biner ini untuk komputer dengan arsitektur i386, yaitu komputer yang memakai mikroprosesor Intel 386 ke atas. Kemudian semua paket adalah versi rpm yang pertama. *Software* *dhcp* mempunyai versi 3.0.1rc9 dan *pdmnu* mempunyai versi 1.2.77.

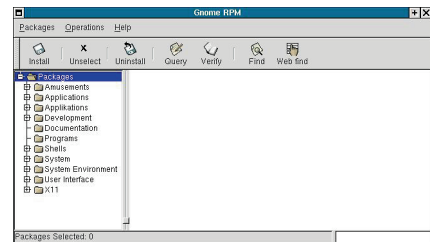
Dalam sistem penamaan ini terdapat dua versi, yaitu versi *software* dan versi rpm. Karena seringkali *software* yang sama dipaket beberapa kali. Pasti timbul pertanyaan, mengapa mesti dipaket berkali-kali kalau *software*-nya itu-itu juga. Ini ada hubungannya dengan pembuatan file spesifikasi dan peletakan file-file *software* yang bersangkutan. Karena dalam membuat file spesifikasi untuk file rpm mirip sekali dengan membuat program. Setiap kali ada modifikasi file spesifikasi, versi rpm akan berubah. Baca juga artikel Utama lainnya tentang cara membuat RPM.

Mengenal istilah paket

Paket RPM adalah suatu *archive* atau sekumpulan file terkompresi yang berkaitan dengan suatu aplikasi. Mirip dengan ZIP dan TAR. Paket juga berisi program/*script* kontrol. *Script* kontrol ini akan dibaca, dieksekusi, dan disimpan oleh program RPM. *Script* inilah yang dijalankan pada proses instalasi, *uninstall*, cek status, dan lain-lain.

Paket RPM dibangun oleh pembuat paket dengan menggunakan *specfile* (*specification file*). *Specfile* ini menunjukkan beberapa hal berikut:

- ➔ Tempat di mana file-file dalam paket akan diinstalasi.
- ➔ Paket-paket lain apa yang diperlukan agar paket berjalan baik.
- ➔ Program apa yang akan dieksekusi setelah atau sebelum instalasi paket.
- ➔ Siapa yang membuat paket, kapan dibuat, dan kegunaan serta isi paket, dan sebagainya.



GNOME-RPM

Agar dapat bekerja dengan efisien, RPM juga menyimpan rekaman semua paket yang telah terinstalasi dalam sistem. Rekaman atau database paket ini biasanya diletakkan dalam direktori `/var/lib/rpm`. Banyak informasi tentang program-program yang sudah terinstalasi ada di database ini.

Yang bisa dilakukan RPM

Banyak hal yang dapat dilakukan dengan RPM, antara lain:

- Melakukan kompilasi dan *patching* kode sumber (*source code*) program. Kompilasi dari paket RPM ini memudahkan para pemula yang masih alergi dengan kompilasi dari *tarball*, karena kompilasi dan *patching* kadang membutuhkan keahlian khusus, seperti memahami fungsi-fungsi file dan direktori, dan lain-lain.
- Mengatasi masalah *dependencies* (ketergantungan antarpaket). Tidak seperti sistem operasi MS Windows, paket di sistem Linux tidak seragam, misalnya dalam hal pemanfaatan sistem dan preferensi yang berbeda antara pengguna satu dengan yang lain. Ketika Anda akan menginstalasi sebuah program baru, RPM akan mengecek apakah semua yang dibutuhkan untuk

menjalankan program baru ini telah ada di sistem. Juga ketika Anda akan menghapus (uninstall) sebuah program, RPM akan mengecek apakah ada program lain yang akan bermasalah jika program tersebut dihapus.

- Menyediakan informasi status sebuah file. Banyak kemungkinan yang diberikan RPM kepada pengguna untuk melihat status suatu paket, yang sudah terinstalasi maupun belum. Anda juga dapat mengecek integritas atau kondisi suatu paket sebelum diinstalasi. Sebelum paket diinstalasi, Anda dapat melihat semua file dan direktori yang akan diubah, berapa ukuran harddisk yang akan dipakai, dan lain-lain.
- Menjaga keamanan. RPM menyediakan mekanisme keamanan dengan PGP atau GPG signature.
- Membuat serba otomatis. RPM dapat digunakan bersama suatu program atau script untuk tugas-tugas perawatan yang bekerja secara otomatis. Telah tersedia program tambahan untuk otomasi ini seperti "rpmfind" dan "urpmi".
- Uninstall yang bersih. Anda hanya cukup memberi satu perintah untuk menghapus paket RPM dari sistem. Perintah ini akan menghapus semua file kecuali file yang telah berubah selama digunakan, misalnya file konfigurasi dan spool (misalnya, e-mail yang ada di `/var/mail/spool` tidak akan dihapus pada saat Anda meng-uninstall paket RPM sendmail).
- Menyimpan file konfigurasi. Ketika Anda meng-upgrade paket, RPM akan menyelamatkan file konfigurasi sebelumnya sebagai file "rpmsave" atau menyimpan file konfigurasi baru sebagai "rpmnew". Anda mendapatkan file konfigurasi baru, tapi tidak kehilangan file konfigurasi yang lama.
- Membedakan antara file "runtime" dan "compile time". Paket RPM berisi library program yang terpisah antara paket "runtime" (paket utama) dan "compile time" (paket 'devel'). Jika tidak melakukan kompilasi sendiri, Anda tidak memerlukan instalasi paket

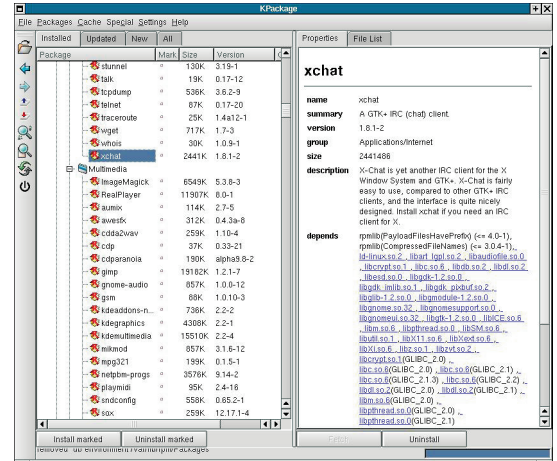
'devel'. Anda hanya membutuhkan paket 'devel' pada saat akan mengompilasi suatu program, lalu menghapusnya setelah kompilasi selesai.

- Mendukung instalasi dari jaringan. Tidak masalah bagi RPM, apakah Anda akan menginstalasi paket dari lokal (harddisk atau CD), atau dari jaringan (NFS, FTP), asalkan Anda menuliskan alamat dan PATH lengkap.
- Mudah dibangun ulang. Setiap paket RPM yang bersifat *Free* atau *Open Source*, menyediakan kode program dalam bentuk RPM (SRPM atau berekstensi `src.rpm`). SRPM berisi semua file yang dibutuhkan untuk membangun paket binari RPM. Ini berguna jika Anda menginginkan program binari yang sesuai dengan sistem (versi kernel, versi library, jenis prosesor, dan lain-lain.).

Yang tidak bisa dilakukan RPM

Seperti semua alat bantu atau tool di dunia komputer, RPM juga memiliki keterbatasan, antara lain:

- Tidak otomatis menyelesaikan masalah ketergantungan (*dependencies*). RPM akan menunjukkan kepada Anda apa yang kurang, tetapi tidak secara otomatis mencarikan dan menginstalasi untuk Anda. Keterbatasan ini dapat diatasi dengan tool tambahan seperti 'urpmi' di Mandrake.
- Tidak ada konfigurasi interaktif. Vendor penyedia paket biasanya menyertakan file konfigurasi dan script untuk dijalankan setelah instalasi. Namun, langkah-langkah yang interaktif harus diberikan secara terpisah bila diperlukan.
- Tidak ada proteksi terhadap kesalahan pemaketan. Kita semua bisa membuat kesalahan, tidak terkecuali para pembuat paket. File-file spesifikasi (*specfiles*) yang tidak benar akan membuat program tidak bekerja



▲ KDE Package Manager

normal setelah diinstalasi. Secara terori, sangat mudah untuk membuat paket RPM dengan tujuan jahat, karena RPM dijalankan oleh *root*. Untuk itu, Anda harus memastikan bahwa paket RPM yang akan diinstalasi dibuat oleh orang atau lembaga yang dapat dipercaya.

- Tidak ada jaminan kompatibilitas. RPM digunakan oleh banyak distro Linux dan personal secara independen. Kebijakan pemaketan yang berbeda menghasilkan cara menempatkan file-file yang berbeda pula. Gunakan paket RPM yang dibuat khusus untuk distro Anda.
- Optimisasi. RPM dapat dioptimisasi dengan cara mengompilasi program sesuai dengan target sistem. Misalnya, Linux Mandrake hanya menyediakan paket RPM yang dibuat untuk prosesor kelas Pentium (i586 ke atas). Biasanya, pembuat paket mencoba memberikan sebanyak mungkin opsi agar sesuai dengan target yang luas, sehingga menghasilkan ukuran paket yang lebih besar. Kata kuncinya, jika Anda ingin *fine tuning*, instalasi program dari source-nya. Baca artikel Utama lainnya tentang menginstalasi dari source.

Instalasi, uninstall, dan upgrade Menginstalasi paket binary

Perintah dasar untuk menginstalasi paket software sangat mudah, yaitu:

```
rpm -i nama_software-versi-versi_rpm.rpm
```


atau

```
rpm --install nama_software-versi-versi_rpm.rpm
```

Contoh:

```
# rpm -i dhcp-3.0.1rc9-1.i386.rpm
```

Sebelum file-file diinstalasi, RPM menjalankan prosedur rutin untuk memeriksa apakah ada file-file yang konflik dengan paket yang lain. Ini biasa terjadi jika paket yang sama dengan versi yang berbeda sudah diinstalasi di dalam sistem. Perhatikan contoh hasil dari perintah di atas jika ada paket software yang sama telah terinstalasi.

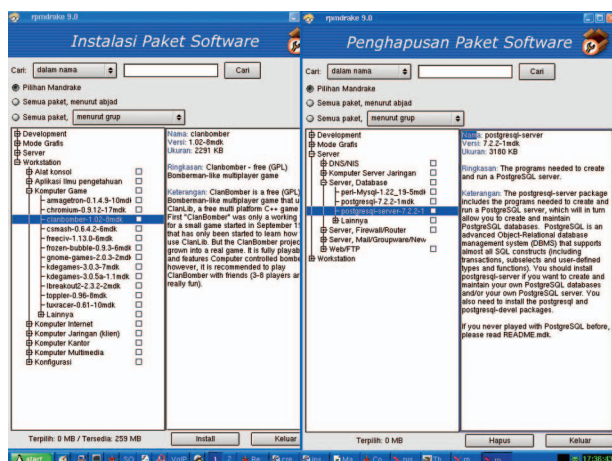
```
package dhcp-2.0pl5-8 (which is newer than dhcp-3.0.1rc9-1) is already installed
```

```
file /usr/sbin/dhcpd from install of dhcp-3.0.1rc9-1 conflicts with file from package dhcp-2.0pl5-8
```

```
file /usr/sbin/dhcrelay from install of dhcp-3.0.1rc9-1 conflicts with file from package dhcp-2.0pl5-8
```

Dari dua pesan terakhir ini jelas bahwa ada dua file yang sama dengan file dalam paket yang akan diinstalasi. Selain pemeriksaan dilakukan terhadap file-file yang konflik, dilakukan juga pemeriksaan ketergantungan paket software. Gunakan option `--nodeps` jika Anda ingin menginstalasi paket software dengan mengabaikan dependensi, dengan risiko program tidak berjalan normal. Misalnya perintah seperti ini:

```
# rpm -ivh --nodeps gnorpm-0.96-11.i386.rpm
```



Software Manager Mandrake 9.0

Menginstalasi paket dari source

Paket source (.src.rpm) harus dibangun ulang (`--rebuild`) sebelum diinstalasi.

```
rpm --rebuild nama_software-versi-versi_src.rpm
```

Hasilnya (.rpm) disimpan pada direktori `/usr/src/RPM/RPMS/arch`. Lalu instalasi dengan perintah `rpm -i` atau `-U`.

Menghapus paket

Paket software dapat dengan mudah dihapus dengan perintah,

```
rpm -e nama_software-versi-versi_rpm
```

atau

```
rpm --erase nama_software-versi-versi_rpm
```

Upgrade/downgrade paket

Proses upgrade software juga mudah dengan menggunakan RPM.

```
rpm -U nama_software-versi-versi_rpm
```

atau

```
rpm --upgrade nama_software-versi-versi_rpm
```

Untuk downgrade paket, gunakan option `--oldpackage`, seperti contoh berikut.

```
# rpm -U --oldpackage pdmenu-1.2.75-1.i386.rpm
```

RPM di X Window

Untuk menjalankan RPM di X Window, KDE menyediakan kpackage (KDE Package Manager), dan GNOME menyediakan gnorpm (GNOME RPM). RPM juga dapat dijalankan melalui *web browser* dengan bantuan Webmin.

RPM di Mandrake

Mandrake memiliki cara tersendiri untuk mengelola paket RPM, yaitu dengan urpm ('User RPM'). Dalam bentuk grafis, Mandrake menyediakan Software Manager (`rpmrake` untuk instalasi dan upgrade, dan `rpmrake-remove` untuk uninstallasi). Urpm bukan pengganti RPM, tapi hanya pelengkap untuk

mempermudah RPM. Beberapa kelebihan urpm adalah sebagai berikut.

- Secara otomatis akan menawarkan instalasi atau uninstall paket yang terkait (*dependencies*). Sedangkan RPM hanya memberi tahu Anda akan adanya dependencies.
- Semua paket yang terkait didata dan diletakkan di mana saja, serta dapat diambil dengan berbagai cara. RPM hanya menyimpan data di lokal.
- Urpm dapat meng-upgrade paket dari Internet dalam proses instalasi jika ditemukan paket yang lebih baru.
- Mendukung instalasi "on demand", update otomatis dan lain-lain.

Apa bedanya antara... RPM dan SRPM?

SRPM tidak berisi program (*binary*) yang sudah dikompilasi, tetapi berisi sumber (*source*) paket RPM yang telah dibuat. SRPM (.src.rpm) diperlukan jika paket RPM yang ada (.rpm) tidak kompatibel dengan library atau PATH di sistem Linux Anda.

Paket.rpm dan Paket-devel.rpm?

RPM devel berisi file-file *header* dan library yang dibutuhkan untuk kompilasi program yang berkaitan dengan paket tersebut. Misalnya, jika Anda ingin membuat program yang jalan di X, Anda harus menginstalasi paket XFree86-devel.

Instalasi dan upgrade?

Upgrade (`-U`) mengganti versi sebelumnya. Sedangkan instalasi (`-i`) tidak mengubah versi yang telah ada (bila mungkin). Misalnya, Anda dapat mempertahankan dua versi suatu paket library, bila kedua versi itu dibutuhkan oleh dua atau lebih program yang berbeda.

File.rpmnew dan File.rpmsave?

File-file ini berkaitan dengan konfigurasi. File `.rpmnew` berisi konfigurasi paket versi baru, konfigurasi aslinya tidak diubah. Sedangkan file `.rpmsave` berisi kopi file konfigurasi paket versi lama. File konfigurasi yang asli diganti dengan file konfigurasi paket versi baru.

Sumber:

<http://www.mandrakeuser.org/docs/basics/>

Membuat Paket RPM

Banyak program baru belum berbentuk RPM, tapi masih berupa *source tarball*. Di lain pihak, banyak pengguna Linux menginginkan RPM. Anda dapat berjasa dengan membuat RPM-nya.

Setelah merasakan kemudahan-kemudahan yang ditawarkan oleh manajemen paket RPM, tentu kita bertanya-tanya apakah sulit membuat paket software dalam format RPM? Jawaban untuk pertanyaan pendek ini bisa bervariasi, tergantung pada seberapa kompleks software yang akan kita paket.

Apa yang perlu kita siapkan untuk membuat paket dengan RPM? Cek daftar berikut:

- Sumber program yang akan kita paket, biasanya dalam format terkompresi nama_file.tar.gz.
- Program RPM, secara default program ini sudah terinstalasi di distribusi semacam RedHat, Mandrake, dan SuSE.
- Paket-paket *development* yang diperlukan untuk proses kompilasi, ini tergantung keperluan software yang akan dipaket.
- File spec yang diperlukan untuk proses pembuatan paket RPM.

Pembahasan kita akan berkisar pada file spec, karena di file inilah tersedia informasi yang diperlukan untuk proses kompilasi, instalasi paket, dan pembuatan paket. Saya memakai distribusi RedHat 7.2 dalam pembahasan ini.

Paket pendukung

Sebelum kita membuat file spec, harus kita yakinkan terlebih dahulu bahwa paket-paket pendukung sudah terinstalasi. Yang pertama adalah program RPM. Jalankan perintah ini.

```
$ rpm -qa | grep rpm
```

Jika keluar tanggapan kurang lebih seperti di bawah ini berarti sudah ada program RPM.

```
rpm-devel-4.0.3-1.03
rpm-4.0.3-1.03
rpm-build-4.0.3-1.03
```

Lihat nama paket rpm-4.0.3-1.03 dan rpm-build-4.0.3-1.03. Jika Anda memakai distribusi RedHat, Mandrake, atau SuSE, pasti sudah terinstalasi, hanya versinya yang mungkin berbeda.

Paket-paket pengembangan software juga harus sudah terinstalasi di sistem. Komponen utama biasanya adalah kompilator, misalnya gcc. Seperti cara di atas, kita bisa memeriksa apakah kompilator bahasa c kita sudah terinstalasi.

```
$ rpm -qa | grep cc
```

Tanggapan yang diharapkan kurang lebih sebagai berikut.

```
gcc-2.96-98
gccgcc-2.96-98-g77-2.96-98
gcc-c++-2.96-98
gcc-objc-2.96-98gcc-2.96-98
```

Lihat nama file gcc-2.96-98. Paket-paket pengembangan software lainnya yang harus terinstalasi tergantung pada software yang akan dipaket. Untuk mengetahui paket apa saja yang diperlukan, cara paling mudah adalah kita kompilasi sesuai dengan petunjuk kompilasi dari dokumentasi kode sumber yang akan kita paket. Jika proses kompilasi secara manual berjalan lancar, tentu saja ketika kita akan membuat paket software, satu langkah sudah pasti tidak bermasalah.

Membuat file spec

Sebagai contoh, kita akan memakai software kompilator bahasa assembler untuk keluarga microcontroller 8051. Dapatkan paket *source code*-nya dari Internet. Nama paket source code dalam contoh ini adalah as31_beta3.tar.gz.

Untuk membuat file spec kita harus mengetahui file-file apa saja yang akan diinstalasi oleh software yang bersangkutan. Informasi ini bisa kita dapatkan dengan beberapa cara, yaitu:

- Dari file dokumentasi paket source code.

- Dari file Makefile paket source code, bisa kita lihat file-file yang akan diinstalasi.
- Dengan cara mengompilasi secara manual dan melihat perbedaan setelah dan sebelum instalasi.

Kita kembali dengan contoh kita, kita coba lihat isi dari file as31_beta3.tar.gz. Jalankan perintah berikut.

```
$ tar xzvf as31_beta3.tar.gz
```

```
as31/
as31/Makefile
as31/README
as31/as31.1
as31/as31.c
as31/as31.h
as31/as31_gtk.c
as31/emitter.c
as31/lexer.c
as31/parser.y
as31/run.c
as31/symbol.c
as31/tests/
as31/tests/extra.asm
as31/tests/paulmon1.asm
as31/tests/paulmon2.asm
as31/tests/paulmon1.ref
as31/tests/paulmon2.ref
as31/tests/extra.ref
```

File as31/README menjelaskan bahwa proses instalasi hanya perlu menjalankan perintah make dan menyalin file-file binary hasil dari proses ini ke direktori */usr/local/bin*.

Selanjutnya kita coba proses kompilasi manual untuk memeriksa apakah ada paket-paket pendukung yang belum diinstalasi.

```
$ cd as31
$ make
```

Proses ini harus berjalan tanpa hambatan. Jika ada software yang diperlukan belum terinstalasi, maka kita harus menginstalasinya terlebih dahulu agar proses kompilasi berjalan lancar.

Dari proses kompilasi ternyata dihasilkan hanya dua file binary, yaitu as31 dan as31_gtk. File-file ini akan kita salin ke direktori */usr/local/bin*.

Salin file as31_beta3.tar.gz ke direktori */usr/src/redhat/SOURCES/*.

```
# cp as31_beta3.tar.gz /usr/src/redhat/
SOURCES/
```

Jika Anda memakai distribusi Slackware direktori ini adalah `/usr/src/rpm/SOURCES/`. Untuk distribusi yang lain bisa Anda lihat dengan perintah `rpm -ql` dari paket `rpm-build`. Struktur direktori untuk membangun paket RPM pada distribusi RedHat seperti ini.

```
/usr/src/redhat/
├── BUILD
├── RPMS
│   ├── athlon
│   ├── i386
│   ├── i486
│   ├── i586
│   ├── i686
│   └── noarch
├── SOURCES
├── SPECS
└── SRPMS
```

Sekarang waktunya kita membuat file spec-nya. Lihat contoh file spec berikut.

```
Summary: Assembler untuk Mikrokontroler
keluarga 8051.
Name: as31
Packager: Salman AS <sas@salman.or.id>
Version: beta3
Release: 1
Url: http://www.pjrc.com/tech/8051/
Source: as31_beta3.tar.gz
Copyright: GPL
Group: Development/Languages
Buildroot: /tmp
%description
Assembler source code program assembly
untuk keluarga mikrokontroler 8051.
%prep
%setup -n as31 -q
%build
make
%install
mkdir -p %buildroot/usr/local/bin/
install as31 %buildroot/usr/local/bin/
install as31_gtk %buildroot/usr/local/bin/
%files
/usr/local/bin/as31
/usr/local/bin/as31_gtk
%doc README tests/*
```

Simpan file spec ini di direktori `/usr/src/redhat/SPECS/`. Kemudian jalankan

```
rpm dengan option -ba, seperti ini:
# rpm -ba /usr/src/redhat/SPECS/as31.spec
```

Bagian awal dari spec file disebut Preamble.

```
Summary: Assembler untuk Mikrokontroler
keluarga 8051.
Name: as31
Packager: Salman AS <sas@salman.or.id>
Version: beta3
Release: 1
Url: http://www.pjrc.com/tech/8051/
Source: as31_beta3.tar.gz
Copyright: GPL
Group: Development/Languages
Buildroot: /tmp
%description
Assembler source code program assembly
untuk keluarga mikrokontroler 8051.
```

Bagian ini berisi informasi-informasi yang akan didapatkan oleh user jika dia melakukan perintah `rpm -q`. *Direktif Name, Version, dan Release* nantinya akan dipakai menjadi nama file paket RPM yang dihasilkan. *Version* adalah versi software. Sedangkan *Release* adalah versi dari paket RPM yang dihasilkan. *Group* akan dipakai untuk menentukan klasifikasi paket software yang akan kita buat. Sedangkan *Buildroot* adalah direktori di mana proses pembangunan paket akan dijalankan. Kemudian sisa *Direktif* yang lain adalah informasi yang telah jelas dari makna direktifnya sendiri.

Bagian selanjutnya adalah *Prep* atau persiapan. Perintah-perintah di bagian ini akan dijalankan sebelum proses kompilasi dijalankan.

```
%prep
%setup -n as31 -q
```

Dalam bagian ini disediakan macro `%setup` yang akan mengurai paket source code dalam format `tar.gz` dan pindah ke direktori hasil proses penguraian file tersebut. Makro di atas maksudnya adalah source code jika diurai hasilnya terdapat dalam direktori `as31`. Nama ini kita dapatkan jika kita mengurai paket secara manual dengan perintah `tar`. Option `-q` berfungsi untuk melakukan proses dengan menampilkan pesan

seminimal mungkin. Untuk mengetahui proses pelaksanaan bagian *Prep* ini, jalankan perintah berikut.

```
# rpm -bp /usr/src/redhat/SPECS/as31.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-
tmp.1343
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd /usr/src/redhat/BUILD
+ rm -rf as31
+ tar -xf -
+ /bin/gzip -dc /usr/src/redhat/SOURCES/
as31_beta3.tar.gz
+ STATUS=0
+ '[' 0 -ne 0 ']'
+ cd as31
++ /usr/bin/id -u
+ '[' 0 = 0 ']'
+ /bin/chown -Rhf root .
++ /usr/bin/id -u
+ '[' 0 = 0 ']'
+ /bin/chgrp -Rhf root .
+ /bin/chmod -Rf a+rX,g-w,o-w .
+ exit 0
```

Dari output perintah di atas dapat dilihat bahwa makro `%setup` paling tidak menjalankan perintah:

1. Masuk ke direktori `/usr/src/redhat/BUILD`.
2. Menghapus direktori source code jika sebelumnya sudah ada direktori yang mempunyai nama yang sama dengan paket yang akan dikompilasi.
3. Mengurai paket source code dari direktori `/usr/src/redhat/SOURCES/`, nama file diambil dari direktif *Source* dalam bagian *Preamble*.
4. Pindah direktori ke direktori kode sumber software.
5. Mengubah kepemilikan, group, dan hak akses dari file dan direktori dalam source code.

Bagian selanjutnya adalah *Build*. Dalam bagian ini proses kompilasi akan dijalankan.

```
%build
make
```

Isikan perintah-perintah shell yang diperlukan di sini. Perintah yang kita perlukan untuk contoh kita hanya satu perintah saja, yaitu `make` yang dijalankan dalam direktori utama kode sumber.

Untuk menjalankan semua perintah dalam file spec sampai tahap kompilasi bisa kita jalankan perintah:

```
# rpm -bc /usr/src/redhat/SPECS/as31.spec
```

Bagian *Install* akan menjalankan proses instalasi software hasil kompilasi. Biasanya di bagian ini berisi satu perintah, *make install*. Tapi karena paket yang kita contohkan di sini tidak tersedia fasilitas ini, tapi dari dokumentasinya diketahui instalasi hanya menyalin dua file ke direktori */usr/local/bin/*, maka bisa kita berikan perintah instalasi langsung.

```
%install
mkdir -p %buildroot/usr/local/bin/
install as31 %buildroot/usr/local/bin/
install as31_gtk %buildroot/usr/local/bin/
```

String *%buildroot* akan diganti dengan */tmp*, seperti disebutkan dalam bagian Preamble. Jadi perintah di atas, bisa dijelaskan kira-kira seperti ini,

1. Buat direktori di */tmp/usr/local/bin/*.
2. Instalasi file *as31* dan *as31_gtk* ke direktori */tmp/usr/local/bin/*.

Jika kita hanya ingin menjalankan perintah sampai tahap instalasi saja, tidak

dilanjutkan dengan pembuatan paket, perintah yang diperlukan adalah:

```
# rpm -bi /usr/src/redhat/SPECS/as31.spec
```

Langkah terakhir adalah menentukan file-file apa saja yang akan dimasukkan ke dalam paket RPM beserta dokumentasinya. Informasi ini disebutkan dalam bagian files dan doc. Lihat cuplikan berikut.

```
%files
/usr/local/bin/as31
/usr/local/bin/as31_gtk
%doc README tests/*
```

Dari contoh ini file yang akan dimasukkan ke dalam paket RPM adalah *as31* dan *as31_gtk* yang terdapat dalam direktori */usr/local/bin/*. Tentu saja file tersebut diambil dari path relatif */tmp* yang disebutkan di *%buildroot*. Dan dokumentasi diambil dari file *README* dan semua file dari direktori *tests*.

Membangun paket binari dan source

Untuk membangun paket dalam format binari jalankan perintah ini:

```
# rpm -bb /usr/src/redhat/SPECS/as31.spec
```

Untuk membangun paket dalam format kode sumber, perintahnya:

```
# rpm -bs /usr/src/redhat/SPECS/as31.spec
```

Sedangkan untuk membangun keduanya, perintahnya:

```
# rpm -ba /usr/src/redhat/SPECS/as31.spec
```

Hasil dari ketiga perintah di atas adalah file paket RPM dalam format binari terdapat di direktori */usr/src/redhat/RPMS/i386/* dan paket RPM dalam format kode sumber di direktori */usr/src/redhat/SRPMS/*.

Sebagai penutup untuk mengujinya, Anda bisa menginstalasi paket software binari dengan perintah ini:

```
# rpm -ivh /usr/src/redhat/RPMS/i386/as31-beta3-1.i386.rpm
```

dan untuk menginstalasi paket dalam format source code:

```
# rpm -ivh /usr/src/redhat/SRPMS/as31-beta3-1.src.rpm
```

Anda juga bisa membangun ulang paket dengan perintah:

```
# rpm --rebuild /usr/src/redhat/SRPMS/as31-beta3-1.src.rpm
```

Instalasi Program Bukan RPM

RPM tidak selalu menyenangkan. Kompilasi dari *source* juga bisa membuat *stress*. Beruntunglah, ada pembuat program yang memberikan cara seperti yang Anda temukan di dunia Windows.

Beberapa perusahaan pembuat software tidak menggunakan format RPM, tetapi *tar.gz* atau *tarball*. Alasannya sederhana, tidak semua distro Linux menggunakan RPM. Misalnya, Adobe dengan Acrobat Reader-nya, Netscape (web browser, mail, dan lain-lain.), Macromedia (Flash), dan Real Network (Real Player). Agar Anda tidak tersesat, membaca petunjuk instalasi (README atau INSTALL) menjadi penting.

Mendapatkan dan mengekstrak paket

Setelah Anda menemukan paket program yang akan diinstalasi, simpan ke suatu direktori yang mudah dikenali, misalnya di

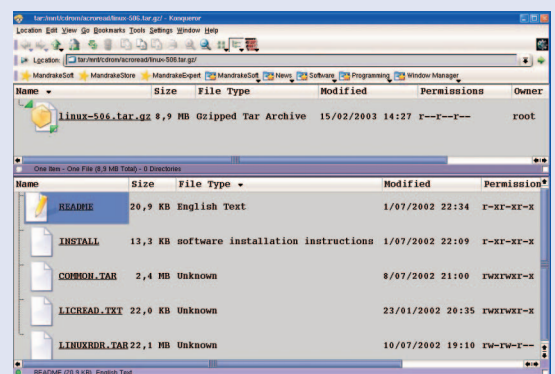
bawah direktori kerja (*home*) Anda. Beberapa *web browser* menyediakan fasilitas memilih direktori untuk menyimpan file hasil *download*. Jika paket program ada di CD, sebaiknya *copy* ke direktori kerja, meskipun Anda dapat mengekstrak langsung dari direktori CD.

Cara mudah mengekstrak bagi pemula adalah menggunakan *file manager* di KDE atau GNOME. Anda tinggal klik nama paket. Lihat contoh Koqueror di KDE sedang mengekstrak file Acrobat Reader dari CD *InfoLINUX* edisi April 2002 (Gambar 1).

Namun, cara manual berikut ini lebih "menantang" karena dapat dijalankan tidak harus membutuhkan X Window.

```
tar xzvf nama-paket.tar.gz
tar xzvf nama-paket.tgz
tar xjvf nama-paket.tar.bz2
```

Catatan: Option *xz* pada baris pertama dan kedua untuk mengekstrak paket *tar* dan *gzip*. Option *xj* pada baris ketiga untuk paket yang dikompres dengan



Gambar 1. Konqueror KDE Mengekstrak Tarball

bzip2. Opsi v untuk menampilkan proses ekstraksi, dan option f untuk menunjuk sebuah nama file (nama-paket.tgz). Semua perintah tar di atas akan menghasilkan file ekstrak di bawah direktori kerja saat ini (*current directory*).

Perintah tar ini punya banyak opsi. Sebaiknya Anda jalankan dengan opsi t (hanya untuk tes atau melihat isi file tarball) sebelum x (mengeksktr). Contoh berikut untuk melihat isi file tarball Acrobat Reader dari CD *InfoLINUX* edisi April 2002. Acrobat Reader adalah program untuk membaca file berbentuk PDF (*Portable Document Format*).

```
mount /mnt/cdrom
cd /mnt/cdrom/acroread
tar tzf linux-506.tar.gz
./
./INSTALL
./README
./LICREAD.TXT
./LINUXRDR.TAR
./COMMON.TAR
```

Perintah tar tzf menunjukkan bahwa hasil ekstrak tidak membuat direktori baru, sehingga dapat membingungkan. Tambahan opsi -C /tmp/acroread akan menyimpan hasil ekstraksi di bawah direktori /tmp/acroread. Contoh, kita ingin mengekstrak file tarball Acrobat Reader ke direktori /tmp/acroread.

```
mkdir /tmp/acroread/
tar xzvf linux-506.tar.gz -C /tmp/acroread
ls /tmp/acroread
INSTALL
README
LICREAD.TXT
LINUXRDR.TAR
COMMON.TAR
```

Catatan: Ada paket program yang tersedia dalam format binari tunggal, misalnya beberapa paket Java di CD *InfoLINUX* April 2003, j2sdk-1.4.1-01-linux-i586-gcc2.95.bin. Paket seperti ini berupa file yang dapat mengekstrak dan menginstalasi sendiri (*self-extracting and installing executables*).

Menginstalasi paket

Jika Anda telah memiliki paket program versi sebelumnya, buat *back up* file-file

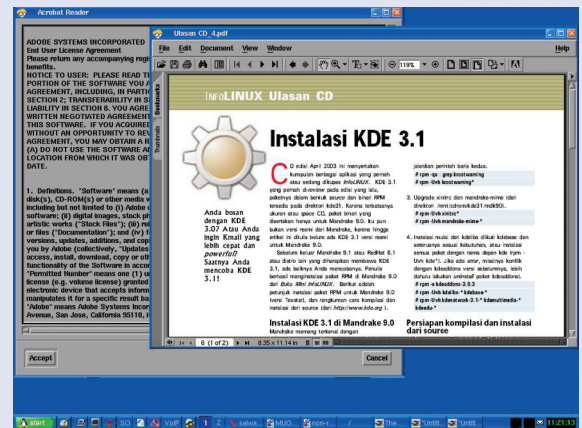
penting. Misalnya, Anda ingin menginstalasi Netscape versi terbaru. Sebelum meng-uninstall versi sebelumnya, copy file *bookmarks.html* yang ada di direktori .netscape ke direktori lain.

Sebelum melakukan instalasi, misalnya Acrobat Reader, baca isi file README lalu baca isi file INSTALL. File README berisi petunjuk instalasi dan cara menggunakannya. File INSTALL berisi script atau program untuk melakukan instalasi.

Untuk menginstalasi program, Anda harus bekerja sebagai root, misalnya dengan menjalankan perintah "su" jika saat ini Anda masih login sebagai user biasa. Langkah-langkah berikut menggambarkan proses instalasi Acrobat Reader:

1. Ekstrak file tarball ke direktori /tmp/acroread (lihat contoh sebelumnya), lalu jalankan su.
\$ su
password:
cd /tmp/acroread
./INSTALL
2. Tekan tombol [space] untuk membaca atau melewati tampilan lisensi, lalu ketik "accept" sebagai tanda setuju.
3. Jawab pertanyaan yang muncul atau tekan [enter] jika setuju dengan jawaban yang disodorkan. Misalnya, Acrobat Reader akan diinstalasi di direktori /usr/local/Acrobat5. Jika direktori tersebut belum ada, Anda diminta menjawab "y" atau langsung [enter] untuk mulai instalasi.
4. Buat shortcut atau soft-link (ln -s) file script untuk menjalankan program ini ke direktori /usr/bin agar langsung dapat dijalankan tanpa menuliskan PATH-nya.
ln -s /usr/local/Acrobat5/bin/acroread /usr/bin/

5. Coba jalankan program dengan mengetikkan di terminal yang ada di desktop (Acrobat Reader harus dijalankan dari X Window). Klik "Accept" lalu buka file PDF. Sebagai contoh, Acrobat Reader membuka file PDF yang dibuat oleh bagian layout



▲ Gambar 2. Acrobat Reader membuka file PDF

InfoLINUX, seperti terlihat dalam Gambar 2.

```
# acroread & (sebagai root)
```

```
$ acroread & (sebagai user biasa)
```

6. Untuk menghemat ruang harddisk, hapus file-file hasil ekstrak.

```
# cd /tmp
# rm -fr acroread
```

Menjalankan program

Jika file binari tidak dapat dieksekusi, bisa jadi karena belum ada permisi "x" terhadap file tersebut. Sebagai contoh, jalankan perintah ini agar semua user bisa menjalankan Acrobat Reader.

```
# chmod a+x /usr/local/Acrobat5/bin/acroread
```

Biasanya, proses instalasi dari tarball atau bukan RPM, tidak meletakkan file binari (*executable*) ke direktori PATH standar (/bin, /usr/bin, dan /usr/local/bin). Dalam contoh Acrobat Reader, file binari acroread berada pada /usr/local/Acrobat5/bin. Selain dengan cara membuat soft-link, Anda juga dapat menjalankan dengan menuliskan PATH secara lengkap.

```
$ /usr/local/Acrobat5/bin/acroread &
```

Cara seperti ini kadang kurang menyenangkan untuk pengguna awam. Mandrake menyediakan Menudrake untuk meletakkan program ini ke dalam menunya. KDE juga memiliki cara yang mudah untuk membuat ikon di *desktop* untuk menjalankan program.

Sumber:

<http://www.mandrakeuser.org/docs/basics/bnorpmp.html>

DPKG dan APT

Distro yang satu ini mungkin kurang populer di negara kita. Akan tetapi, *packet management* yang dimiliki dan digunakan oleh distro ini pantas diberikan acungan jempol.

Kita akan membahas sekilas bagaimana struktur paket *deb* sampai kepada berbagai *frontend* yang digunakan untuk memanipulasi paket tersebut. Tidak ketinggalan, bagi Anda yang senang mencoba-coba, kami sertakan pula cara penggunaan dan pembuatan paket *debian*.

Setiap paket *debian* diberi nama file dengan ekstensi *.deb*. Sama seperti halnya *packet management* lain, sebuah software dapat dipisahkan menjadi beberapa paket. Hal ini dimaksudkan agar pengguna tidak perlu menginstalasi keseluruhan software hanya untuk menggunakan program tersebut. Format file *.deb* yang digunakan saat ini telah digunakan sejak *Debian* versi 0.93.

Sederhananya, paket *debian* adalah sebuah archive *ar* dengan magic number `!<arch>`. Isi paket *debian* dapat dibuka dengan mudah apabila Anda memiliki *ar*, yang merupakan standar di setiap distro. Dapat lebih mudah dibuka lagi apalagi *Midnight Commander* Anda dikonfigur untuk mampu membaca format *.deb*. Enter saja di atas nama file-nya, dan seluruh isinya akan kelihatan dengan gamblang.

Mari sejenak melihat isi sebuah paket *debian* untuk program *gzip*, sebuah tool kompresi. Pada *debian* 3.0r0 untuk arsitektur *i386*, nama filenya adalah *gzip_1.3.2-3_i386.deb*. Berikut ini adalah isi dari level pertama direktorinya:

+ CONTENTS
+ DEBIAN
-INFO
-INSTALL

Direktori *CONTENTS* berisikan isi fisik file sistem yang akan di-copy-kan begitu paket ini diinstalasi. Sedangkan direktori *DEBIAN* berisikan hal-hal yang dibutuhkan dalam pemrosesan isi paket.

Kita akan melihat secara mendetail isi direktori ini.

Kita memasuki direktori *DEBIAN*. Pada paket *gzip* kita akan menemukan lima buah file, yaitu *control*, *md5sums*, *postinst*, *preinst* dan *prerm*. Lantas apakah kegunaan dari masing-masing file tersebut?

Melangkah lebih lanjut, kita akan melihat isi dari file *control*. File ini akan mengontrol tindakan yang akan dikenakan terhadap paket ini. Bisa dikatakan, file ini adalah jantung dari sebuah paket. Dari isi file ini, kita akan mendapatkan nama paket, prioritas, *section*, *installed size*, *maintainer*, arsitektur, versi, ketergantungan akan paket lain, dan deskripsi singkat. Mari kita intip isi dari file *control* untuk paket *gzip*.

Package: gzip
Version: 1.3.2-3
Section: base
Priority: required
Architecture: i386
Essential: yes
Pre-Depends: libc6 (>= 2.2.4-4)
Depends: debiutils (>= 1.6)
Installed-Size: 192
Maintainer: Bdale Garbee <bdale@gag.com>
Description: The GNU compression utility. This is the standard GNU file compression utility, which is also the default compression tool for Debian. It typically operates on files with names ending in '.gz'.
.
This package can also decompress '.Z' files created with 'compress'.

Keluar sementara dari file *control*, kita akan melihat kegunaan dari file *md5sums*. File ini berisikan daftar dari *md5sum* dari setiap file yang akan di-copy-kan ke file sistem.

File *postinst*, *preinst*, dan *prerm* lain lagi. Mereka merupakan bagian dari himpunan script yang akan dijalankan

sesuai dengan nama script masing-masing.

Script *preinst* akan berjalan sebelum isi paket tersebut diuraikan dari paketnya. Umumnya, fungsi script ini adalah menghentikan servis yang sedang berjalan, terutama apabila upgrade sedang dilakukan.

Script *postinst* akan berjalan setelah isi paket diuraikan. Umumnya file ini bertugas untuk menyusun nilai-nilai yang dibutuhkan untuk file konfigurasi. Singkat kata, script ini bertugas untuk menyempurnakan instalasi suatu paket.

Script *prerm* dan *postrm* lain lagi. Script *prerm* dijalankan sebelum penghapusan file yang diasosiasikan pada paket *debian*, sementara script *postrm* dijalankan untuk memodifikasi link atau file lain yang berhubungan dengan paket tersebut.

Paket *debian* disusun dengan sedemikian rupa sehingga pemrosesan suatu paket tidak akan berakibat tidak baik pada sistem yang sedang berjalan. Sejenak, alihkan perhatian Anda kembali kepada file *control*.

Pada file *control*, kita akan melihat *field Section*. *Field section* dibutuhkan karena pada *Debian* 3.0r0 saja, jumlah paket telah mencapai angka di atas 8000 paket. *Section-section* yang ada, di antaranya *admin*, *base*, *devel*, *doc*, *editors*, *games*, *graphics*, *interpreters*, *libs*, *mail*, *math*, *misc*, *net*, *news*, *non-US*, *oldlibs*, *othersofs*, *science*, *shells*, *sound*, *tex*, *text*, *utils*, *web*, dan *x11*. Masing-masing isi dari *section* tersebut dapat di browse di <http://packages.debian.org/>.

Kembali ke file *control*, rasa-rasanya *field priority* (prioritas) cukup menarik perhatian. Paket-paket *debian* umumnya dapat dibagi dalam prioritasnya pula. Terdapat lima tingkat kepentingan yaitu:

- **required.** Paket yang memiliki tingkat kepentingan ini sangat dibutuhkan oleh sistem yang berjalan aktif dan produktif. Penghapusan paket dengan prioritas ini bisa berujung pada tidak bekerjanya sistem.
- **Important.** Paket yang memiliki tingkat kepentingan ini seharusnya terdapat pada sistem GNU/Linux yang umum.

- *Standard*. Paket dengan tingkat kepentingan ini umumnya dapat dengan mudah ditemukan pada sistem GNU/Linux normal.
- *Optional*. Paket ini memiliki tingkat kepentingan yang cukup rendah, akan tetapi umumnya dapat meningkatkan kepuasan menggunakan GNU/Linux apabila diinstalasi. Contohnya adalah X.
- *Extra*. Paket dengan tingkat kepentingan seperti ini cukup jarang ditemukan. Umumnya karena tidak cocok untuk tingkat kepentingan lainnya.

Mari kembali melongok kepada jantung paket, file control. *Field Depends* menunjukkan bahwa paket gzip tergantung pada paket lainnya. Selain depends, paket Debian juga mengenal *recommends*, *suggests*, *conflicts*, *replaces*, *provides*. A Recommends B berarti sang *packet maintainer* berpendapat bahwa umumnya pengguna sebaiknya menginstalasi B selain A. Kondisi A *suggests* B tercapai karena umumnya paket B dapat meningkatkan kinerja paket A. Sementara kondisi A *conflicts* B terjadi apabila paket A tidak akan diinstalasi apabila terdapat paket B pada sistem. A *replaces* B menunjukkan bahwa instalasi paket A akan menggantikan paket B yang telah terinstalasi. Sementara A *provides* B menyatakan bahwa paket A telah berisikan fungsionalitas paket B.

Kembali ke permasalahan awal tentang sulitnya instalasi program, Debian GNU/Linux ingin meyakinkan bahwa penggunaanya akan benar-benar mendapatkan apa yang diinginkan tanpa mengorbankan sistem yang sedang berjalan. Walaupun untuk itu, sang maintainer junior harus mati-matian membaca Debian Packaging Manual. Tidak salah kalau Debian mengklaim dirinya memiliki sistem pemaketan yang terbaik, secara praktikal.

Semua kerumitan tersebut seakan berkurang oleh program dpkg dan semakin tidak terasa apabila APT turut serta. Dengan nama file `gzip_1.3.2-3 i386.deb`, berikut ini kita akan

melihat sekilas cara penggunaan program dpkg.

```
dpkg -i <package.deb>
```

Melakukan instalasi untuk sebuah paket debian

```
dpkg -c <package.deb>
```

Menampilkan isi dari suatu paket debian

```
dpkg -l <package.deb>
```

Menampilkan informasi dari suatu paket debian

```
dpkg -r <package>
```

Menghapus paket yang telah terinstalasi

```
dpkg -P <package>
```

Melakukan purge dari paket yang telah terinstalasi. Beda antara menghapus dan melakukan purge adalah ketika menghapus paket hanya akan menyebabkan data dan program terhapus, sementara purge akan menyebabkan keseluruhan isi paket terhapus, termasuk file konfigurasinya. Dengan demikian, ketika Anda ingin melakukan penghapusan yang bersih, gunakan selalu opsi untuk melakukan purge.

```
dpkg -L <package>
```

Mendaftar file yang terinstalasi dari paket debian.

```
dpkg -s <package>
```

Menampilkan informasi dari suatu paket yang terinstalasi.

dpkg-reconfigure <package>

Melakukan konfigurasi ulang suatu paket, apabila paket tersebut menggunakan debconf, yang menyediakan antar muka konfigurasi yang konsisten untuk instalasi paket.

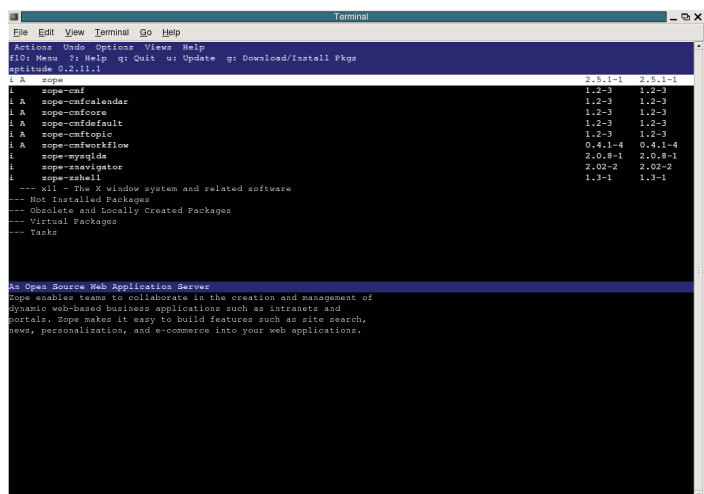
Sejenak, walaupun paket debian telah dibuat dengan hati-hati, penggunaan program dpkg terkesan sama dengan packet management lain seperti rpm yang (umumnya) berorientasi pada file paket itu sendiri.

Dalam artian, opsi -i yang berguna untuk melakukan instalasi paket memerlukan kehadiran paket debian tersebut di file sistem dalam wujud file berekstensi .deb.

Lantas, bagaimana kalau Anda ingin melakukan instalasi suatu paket sementara informasi satu-satunya yang Anda miliki hanya nama paket tersebut? Anda lupa segalanya, termasuk nama file paket Debian yang termasuk panjang. Dpkg sendiri pun tidak bisa bertanggung jawab penuh untuk melakukan instalasi otomatis paket-paket yang dibutuhkan oleh paket yang ingin diinstalasi. Barangkali Anda akan bertanya di mana letak kehebatan sebuah dpkg.

Untuk mengatasi segala permasalahan, APT pun diperkenalkan. APT singkatan dari *Advanced Package Tool* yang akan memaksimalkan kinerja dari dpkg. Untuk melakukan instalasi suatu paket, informasi yang Anda butuhkan hanyalah nama paket tersebut. Bahwa paket tersebut memerlukan 100 paket lainnya, biarkan APT yang menginstalasinya untuk Anda. Lokasinya pun akan diberitahukan oleh APT. APT akan meminta nomor CD di mana suatu paket tersimpan dan akan mencari secara otomatis apabila lokasi paket adalah di Internet.

Bicara soal lokasi, APT bekerja dengan membaca `/etc/apt/sources.list`. Dalam artian, Anda harus menentukan lokasi suatu paket. APT tidak bekerja



🔥 **Mempermudah manaiemen paket debian dengan aptitude**


dengan file paket debian. Umumnya, apabila Anda melakukan instalasi dari CD-ROM, maka *sources.list* Anda akan berisikan daftar CD-ROM-CD-ROM. Anda bisa menambahkan atau mengurangi isi dari *sources.list* menggunakan teks editor standar.

APT sendiri adalah sekumpulan tool. Umumnya Anda akan membutuhkan *apt-get*, yang sudah memiliki tanggung jawab instalasi dan penghapusan paket. Bagi Anda yang bekerja dengan CD-ROM, pastikan *apt-cdrom* juga terinstalasi.

Berikut ini adalah beberapa cara penggunaan *apt-get*:

apt-get install vim	Melakukan instalasi paket vim (dan paket yang dibutuhkan oleh vim)
apt-get remove vim	Menghapus paket vim
apt-get --purge remove vim	Menghapus vim dan konfigurasinya
apt-get update	Memperbarui package list. Lakukan perintah ini setiap kali Anda mengubah <i>sources.list</i>
apt-get upgrade	Melakukan upgrade pada sistem. Apabila terdapat versi baru, maka secara default, <i>apt-get</i> akan melakukan upgrade.

Semua perintah tersebut dapat diberikan di direktori mana saja. APT tidak bekerja dengan file paket debian. Perintah tersebut masih cukup merepotkan bagi Anda? Install *aptitude*, *synaptic*, *KPackage*, atau *gnome-apt* dan semua manajemen paket dapat dilakukan menggunakan program-program tersebut. Manajemen paket semudah klik tombol mouse.

Mungkin Anda akan bertanya. Haruskah saya menggunakan Debian untuk mendapatkan kemampuan APT? Jawabannya adalah Tidak. APT telah di-port ke berbagai distro, diantaranya Conectiva dan Mandrake. 

Instalasi dari Source Code

Instalasi paket binari memang mudah, namun tidak selalu sesuai dengan sistem sehingga program tidak bekerja optimal. Kadang, paket yang akan diinstalasi hanya tersedia dalam bentuk *source*. Solusi terbaik untuk itu semua adalah kompilasi.

Anda tidak harus menguasai bahasa pemrograman kalau hanya ingin mengompilasi paket. Jika saat instalasi sistem Linux kita menginstalasi paket-paket untuk development dan *library-library*-nya, tidak akan ada kesulitan yang besar untuk mengompilasi dan menginstalasi paket-paket *source code*. Kesulitan-kesulitan yang timbul juga bisa diselesaikan jika kita mempunyai sedikit modal kesabaran dan kemauan untuk belajar.

Untuk mempermudah pembahasan, saya akan memakai proses instalasi dari kode sumber paket web server Apache.

Kebutuhan sistem

Apa yang diperlukan untuk menjalankan proses kompilasi dan instalasi? Tentu saja yang pertama adalah kompiler C/C++ karena *source code* ditulis dengan bahasa ini. Dalam distribusi Linux, kompiler bahasa C/C++ yang dipakai adalah kompiler GNU C/C++. Untuk memeriksa apakah kompiler ini sudah terinstalasi, jalankan perintah ini:

```
$ whereis gcc
```

Jika hasil perintah ini kira-kira sebagai berikut:

```
gcc: /usr/bin/gcc /usr/share/man/man1/
gcc.1.gz
```

maka Anda boleh senang karena kompiler gcc sudah terinstalasi.

Paket-paket software seperti *tar*, *gzip*, *gunzip*, dan *bunzip2* sangat diperlukan untuk proses penguraian file kode sumber. Software lain yang dibutuhkan adalah file-file *library* dan *header* (termasuk *kernel-header*). Paket selengkapnya apa saja yang dibutuhkan tergantung pada software yang akan diinstalasi. Jadi akan sangat bervariasi. Dan biasanya kesulitan yang paling sering terjadi adalah karena tidak adanya file-file ini.

Yang terakhir, software yang diperlukan adalah paket *automake*,

autoconf, dan *make*. Dua paket pertama dipakai untuk menghasilkan *Makefile*. Sedangkan program *make* menggunakan *Makefile* untuk melakukan kompilasi.

Download dan baca dokumentasi

Baik, karena kita akan mencoba mengompilasi Apache, *download* dulu file software dari <http://httpd.apache.org>. File ini biasanya dalam format *namafilename.tar.gz*. Jika paket yang didapatkan dengan ekstensi *tar.gz* atau *.tgz*, maka Anda harus mengurai paket tersebut dengan program *tar* dan *gunzip*.

```
$ tar -xzf apache_1.3.27.tar.gz
```

Dengan perintah ini program *gunzip* akan dijalankan oleh program *tar* secara otomatis. File *apache_1.3.27.tar.gz* akan diurai menjadi struktur direktori dan file dalam direktori *apache_1.3.27*. Proses seperti ini hampir seragam untuk software yang lain. Karena kecenderungan pemaketan *source code* dengan format yang sama, mengikuti kebiasaan Free Software Foundation dengan proyek GNU-nya.

Tapi jika suatu saat Anda mendapatkan paket dalam bentuk *tar.bz2*, maka Anda memerlukan paket software *bunzip2*.

```
$ tar -xjvf nama_paket.tar.bz2
```

Kedua format paket tersebut, *tar.gz* dan *tar.bz2* hanya berbeda dalam proses kompresi saja.

Setelah kita uraikan paket software, tugas pertama adalah membaca dokumentasi yang disertakan pada setiap kode sumber. File-file penting yang harus dibaca adalah *README*, *INSTALL*, *CHANGES*, *FAQ*, dan kalau Anda peduli baca juga file *LICENSE* atau *COPYING*. Untuk program-program *open source* kedua file ini biasanya berisi GNU General Public Licence atau yang sepadan.

Konfigurasi, kompilasi, dan instalasi

Langkah pertama proses kompilasi adalah konfigurasi atau persiapan kompilasi. Pada umumnya source code mempunyai skrip yang bernama *configure* untuk menjalankan proses pengecekan kebutuhan library. Selain itu, skrip ini juga berfungsi untuk mengatur konfigurasi hasil file binary yang akan kita kompilasi. Misalnya, letak file hasil kompilasi dan sebagainya. Dan ini juga bergantung pada software yang akan dikompilasi. Untuk melihat daftar *option* (opsi) yang bisa dipakai, jalankan perintah berikut.

```
$ ./configure --help
```

Option umum yang ada biasanya:

- `--prefix=PREFIX`, menunjukkan letak file-file yang tidak bergantung kepada arsitektur.
- `--sysconfdir=DIR`, direktori di mana akan diletakkan konfigurasi dari program yang akan dikompilasi.
- `--mandir=DIR`, direktori di mana akan diletakkan *online* manual.
- `--bindir=DIR`, menentukan direktori dimana akan diletakkan program-program yang boleh diakses oleh user biasa.
- `--sbindir=DIR`, menentukan direktori di mana akan diletakkan program-program yang hanya boleh diakses oleh administrator.

Kembali ke contoh kita. Jalankan skrip *configure*.

```
$ cd apache_1.3.27
```

```
$ ./configure
```

```
Configuring for Apache, Version 1.3.27
```

```
+ Warning: Configuring Apache with default settings.
```

```
+ This is probably not what you really want.
```

```
+ Please read the README.configure and INSTALL files
```

```
+ first or at least run './configure --help' for
```

```
+ a compact summary of available options.
```

```
+ using installation path layout: Apache (config.layout)
```

```
Creating Makefile
```

```
Creating Configuration.apaci in src
```

```
Creating Makefile in src
```

```
+ configured for Linux platform
```

```
+ setting C compiler to gcc
```

```
+ setting C pre-processor to gcc -E
```

```
+ checking for system header files
```

```
+ adding selected modules
```

```
+ using system Expat
```

```
+ checking sizeof various data types
```

```
+ doing sanity check on compiler and options
```

```
Creating Makefile in src/support
```

```
Creating Makefile in src/regex
```

```
Creating Makefile in src/os/unix
```

```
Creating Makefile in src/ap
```

```
Creating Makefile in src/main
```

```
Creating Makefile in src/modules/standard
```

Kalau sampai tahap ini tidak ada masalah, maka Anda sudah setengah jalan. Tapi jika dalam proses ini ada pesan-pesan kesalahan atau peringatan, Anda harus menyelesaikannya terlebih dahulu. Sumber utama pesan-pesan tersebut adalah kurangnya library dan file-file header. Dan masalah ini harus diselesaikan per kasus karena sangat bervariasi.

Setelah proses konfigurasi selesai tanpa kesalahan, kompilasi siap dimulai dengan perintah *make*.

```
$ make
```

Tergantung kecepatan prosesor Anda, proses ini memakan waktu yang cepat jika Anda memakai generasi komputer yang baru. Tapi jika Anda memakai prosesor generasi lama, maka silakan ditinggal untuk *break* atau makan dahulu.

Langkah terakhir adalah proses instalasinya sendiri, "make install", dan pembersihannya, "make clean". Anda harus login sebagai user root agar bisa menjalankan proses instalasi, karena biasanya proses ini akan menyalin file-file ke lokasi yang hanya boleh diakses oleh root.

```
$ su
```

```
# make install
```

```
# make clean
```

Uninstall atau menghapus paket

Untuk mengetahui file-file apa saja yang diinstalasi ke dalam sistem, bisa kita bandingkan perubahan yang terjadi sebelum dan setelah proses instalasi dilakukan. Jalankan perintah ini sebelum menjalankan *make install*.

```
# find / * > file.awal
```

Jalankan perintah serupa setelah *make install*.

```
# find / * > file.akhir
```

Bandingkan kedua file tersebut untuk mengetahui daftar file yang diinstalasi ke dalam sistem. Jadi jika ingin menghapus file-file software dari sistem, Anda bisa menghapus file-file yang ada dalam daftar tersebut. Berikut perintah untuk mengetahui file-file apa saja yang diinstalasi.

```
# diff file.awal file.akhir > file.diff
```

Tentu saja dalam *file.diff* ini ada file-file lain yang tidak termasuk file yang diinstalasi oleh paket terakhir, tapi Anda akan segera tahu mana yang termasuk file yang diinstalasi dan mana yang bukan setelah melihat isi file-nya.

Jika paket source diinstalasi dengan perintah "make install", Anda dapat meng-uninstall-nya dengan lebih dahulu masuk ke direktori yang ada file *Makefile*, lalu menjalankan perintah berikut:

```
# make uninstall
```

Gagal karena masalah library

Masalah yang biasa muncul dalam proses "configure" dan "make" adalah tidak adanya suatu library (lib). Jalankan "make distclean" untuk menghapus efek dari "configure" dan "make" yang gagal, lalu instalasi library yang dibutuhkan. Jika perintah "make distclean" tidak berhasil membersihkan direktori source, jalankan perintah berikut:

```
$ make clean && rm config.cache
```

Jika dalam proses instalasi suatu paket menempatkan library dalam sebuah direktori, misalnya */usr/local/lib*, Anda harus menuliskan nama direktori ini ke dalam file */etc/ld.so.conf*, lalu menjalankan perintah *ldconfig* (sebagai root).

```
# ldconfig
```

Ulangi perintah *configure* dan *make*, hingga tidak ada lagi pesan kesalahan.