Mendeteksi Distribusi Linux yang Digunakan

Terdapat banyak sekali distribusi Linux yang ada di dunia ini. Setiap harinya, distro baru bahkan bisa lahir. Tidak semua distribusi Linux tersebut menuruti standar. Lantas, kalau kita ingin mendeteksi distribusi Linux yang digunakan, apa saja yang bisa dilakukan?

Mendeteksi distro memang pekerjaan mudah, namun sekaligus susah. Mudah karena kita umumnya, kita hanya perlu membaca sebuah file. Susah karena, terkadang kita tidak tahu file apa yang harus dibaca. Ini serius.

Padahal, terkadang, kita perlu mengetahui distro apa yang digunakan. Sebagai contoh, ketika kita menerapkan suatu solusi pada lingkungan multidistro. Sementara, solusi yang kita *deploy* sangatlah dekat dengan sistem/distro. Maka, mengetahui distro apa yang digunakan, versi berapa distro tersebut menjadi penting. Salah mengenali bisa mengakibatkan kita menerapkan solusi yang kurang tepat.

Urusan mendeteksi di Linux sebenarnya relatif mudah dibandingkan dengan sistem operasi lain. Untuk segala sesuatu yang berhubungan dengan kernel misalnya, file-file /proc file system sudah banyak membantu kita.

Sayangnya, nama distro yang digunakan tidaklah menjadi bagian dari internal kernel. Kita tidak bisa mengetahui distro apa yang digunakan hanya dengan membaca salah satu file di /proc misalnya. Kalaupun ada distro yang menempatkan namanya di kernel, itu pun tidak lazim diikuti distribusi lainnya.

Untungnya, *developer* distro umumnya menuliskan nama distro, sekaligus versi distro, pada salah satu file di /etc/. Hanya, sayangnya, nama file yang digunakan berbeda-beda.

- openSUSE 10.2: file /etc/SuSE-release.
- Debian GNU/Linux 4.0: file /etc/debian_version.
- Slackware Linux 11.0: file /etc/slackware-version.
- Fedora: file /etc/fedora-release.

Inilah yang dimaksudkan sebagai mudah tapi susah pada awal tulisan. Apabila kita tahu file apa yang harus dibaca, maka kita langsung mendapatkan distro apa yang digunakan. Namun, dengan penggunaan file yang berbeda-beda oleh setiap distro, maka kita harus memeriksa lebih jauh.

Di tulisan ini, kita akan membangun sebuah *shell script* yang akan memeriksa distro apa yang digunakan. Shell script ini bisa di-*embed* ke dalam solusi Anda, terutama yang berhubungan dekat dengan sistem atau distro.

Berikut ini *source code* distro_detect.sh:

```
#!/bin/sh

#(c) Noprianto, 2007. GPL.

#check 1: detect using known file in
/etc
SIG_FILE=/etc/slackware-version
/etc/SuSE-release /etc/debian_
version
DISTRO_FILE=/dev/null
UNSUPPORTED=0
DISTRO=unknown
VERSION=unknown

for f in $SIG_FILE
do
  if [ -r $f ]
  then
    DISTRO_FILE=$f
    break
  fi
done

case $DISTRO_FILE in
  /etc/slackware-version)
        DISTRO=`cat $DISTRO_FILE|
cut -d' ' -f1`
        VERSION=`cat $DISTRO_FILE|
cut -d' ' -f2`
        ;;
  /etc/SuSE-release)
        DISTRO=`cat $DISTRO_FILE|
head -n1 |  cut -d' ' -f1`
        VERSION=`cat $DISTRO_FILE|
head -n1 | cut -d' ' -f2`
        ;;
  /etc/debian_version)
        DISTRO=debian
        VERSION=`cat $DISTRO_FILE`
        ;;
    *) #check 2: unknown
distribution, check *-release
        UNSUPPORTED=1
        RELEASE_FILE=`ls -1 /etc/
*release 2>/dev/null`
        for f in $RELEASE_FILE
        do
                if [ $f != /etc/
lsb-release ]
                then
                        DISTRO_
FILE=$f
                        break
```

```
                fi
        done
        if [ ! -z $DISTRO_FILE ]
        then
                DISTRO=`head -n1 $DISTRO_FILE`
        fi
        ;;
esac

if [ $UNSUPPORTED -eq 0 ]
then
  echo You are using $DISTRO version $VERSION
  exit 0
else
  echo You are using unsupported distro
  if [ ! -z $DISTRO ]
  then
   echo Distro information: $DISTRO
  fi
  exit 1
fi
```

Berikanlah hak akses *executable* dengan perintah berikut ini:

```
$ chmod +x distro_detect.sh
```

Contoh output di Slackware 11:

```
$ ./distro_detect.sh
You are using Slackware version 11.0.0
```

Contoh output di OpenSUSE 10.2:

```
$ ./distro_detect.sh
You are using openSUSE version 10.2
```

Contoh output di Debian GNU/Linux 4.0:

```
$ ./distro_detect.sh
You are using debian version 4.0
```

Contoh simulasi distro yang tidak didukung, namun memiliki file /etc/system-release (isi file: system version 100):

```
$ ./distro_detect.sh
You are using unsupported distro
Distro information: System version 100
```

Contoh simulasi distro yang tidak didukung, namun file /etc/*release juga tidak ditemukan:

```
$ ./distro_detect.sh
You are using unsupported distro
```

Penjelasan source code:

- Pertama-tama, kita deretkan file penanda unik setiap distribusi di dalam variabel SIG_FILE, dipisahkan spasi setiap file. Setiap file di sini mewakili setiap distribusi yang dikenal.

```
SIG_FILE=/etc/slackware-version /etc/SuSE-release /etc/debian_version
```

- Untuk setiap file di $SIG_FILE, kita akan memeriksa keberadaannya. Apabila ditemukan, maka kita assign nama file tersebut ke variabel $DISTRO_FILE. Perulangan langsung dihentikan.
- Untuk $DISTRO_FILE yang ditemukan, kita secara khusus mendapatkan nama distro dan versi distro, sesuai dengan informasi yang kita ketahui sebelumnya. Misalnya:
  - Slackware 11 menggunakan file /etc/slackware-version, dan nama distro serta versi distro ditulis dalam satu baris dipisahkan spasi.
  - Debian 4.0 menggunakan file /etc/debian_version. Nama distro sudah pasti Debian dan untuk versi distro, kita tinggal baca file /etc/debian_version tersebut.
  - Demikian seterusnya.
  - Khusus untuk yang tidak dikenali:
    - Kita dapatkan file-file /etc/*release.
    - Apabila file bukan /etc/lsb-release (yang berikan informasi Linux Standard Base), maka file pertama yang didapatkan, diasumsikan file yang menyimpan informasi distro. Ini bukanlah cara yang akurat, namun di beberapa sistem dapat berfungsi.
    - Apabila file informasi distro ditemukan, kita membaca baris pertama isi file tersebut dan menganggapnya sebagai informasi distro yang digunakan.
- Terakhir, kita menampilkan informasi distro yang berhasil didapatkan. Khusus untuk distro yang tidak dikenal, apabila pembacaan file /etc/*release membuahkan isi tertentu, barulah kita tampilkan informasinya.

Cara pendeteksian ini masih sangat jauh dari akurat, terutama pada distro yang tidak dikenal. Walau demikian, kita selalu bisa mengembangkan lebih lanjut sampai seakurat mungkin. Misalnya, pada distro yang tidak dikenal, selain memeriksa file /etc/*release, kita juga bisa memeriksa /etc/*_version atau /etc/*-version.

Sampai di sini dulu pembahasan kita. Selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

Menyimpan Format PDF Secara Otomatis

Sering atau selalu mengirimkan *copy* dokumen dalam format PDF? Selalu menggunakan menu File -> Export as PDF? Kerepotan ketika dokumen diubah dan mengirimkan bukan versi terbaru? Gunakan macro untuk mengotomatisasi agar sebuah copy format PDF selalu dibuat ketika dokumen disimpan.

Beberapa perusahaan menerapkan kebijakan untuk selalu mengirimkan file dalam format PDF dan bukannya tetap dalam format OpenDocument OpenOffice.org. Otomatis, Anda pun akan selalu membuat sebuah versi PDF dari dokumen yang akan dikirimkan. Hal ini tidaklah masalah karena OpenOffice.org dapat mengekspor ke format PDF dengan sangat mudah dan baik.

Masalahnya akan muncul kalau Anda termasuk orang yang ceroboh dan mudah lupa. Apalagi kalau dokumen sering dibuka berulang kali dan rentan terhadap perubahan. Ketika Anda lupa untuk mengekspor ke format PDF ketika terakhir menyimpan dokumen, maka Anda berpotensi mengirimkan dokumen versi lama. Hal ini tentu saja bisa menjadi masalah.

Setahu penulis, saat ini belum ada sebuah fasilitas builtin OpenOffice.org yang akan secara otomatis mengekspor ke format PDF ketika Anda menyimpan dokumen. Untungnya, OpenOffice.org mendukung macro, yang dapat digunakan untuk membangun solusi yang belum disediakan.

Di dalam tulisan ini, kita akan membangun sebuah macro sederhana yang akan selalu menyimpan sebuah kopi PDF dengan nama yang sama dengan nama dokumen setiap kali Anda menyimpan. Kopi PDF tersebut akan disimpan tepat setelah dokumen asli selesai disimpan.

Semua contoh di tulisan ini dibangun di atas sistem Slackware 11 dan OpenOffice.org 2.2.0, namun seharusnya dapat diterapkan pada distro lain ataupun OpenOffice.org versi lain. Bahkan, solusi ini seharusnya juga dapat diterapkan pada OpenOffice.org di atas sistem operasi Windows.

Sebelum memasuki macro yang dimaksud, kita akan membahas beberapa hal penting terlebih dahulu: macro sederhana, keamanan, lokasi penyimpanan macro dan *event*.

Macro sederhana 1

Di macro sederhana yang pertama, kita akan menampilkan sebuah pesan. Tujuan dari macro sederhana 1 ini adalah membahas cara membuat macro, mengedit macro, menjalankan macro, dan contoh output dengan *message box*.

Berikut ini adalah langkah-langkah untuk membangun macro sederhana 1:

- Bukalah dokumen baru OpenOffice.org Writer dan simpanlah sebagai macro1.odt. Isi dokumen tidak penting.
- Akseslah menu Tools -> Macros -> Organize Macros -> OpenOffice.org Basic...
- Sebuah dialog OpenOffice.org Basic Macros akan ditampilkan. Pada panel kiri, di bagian Macro from, pilihlah macro1.odt -> Standard.
- Kliklah tombol New. Sebuah dialog New Module akan ditampilkan. Berikan nama Module1 dan kliklah tombol OK.
- Code editor akan ditampilkan, di mana kita akan diberikan kerangka fungsi Main.

```
Sub Main

End Sub
```

- Isikanlah kode berikut ini di antara Sub Main dan End Sub:

```
msgbox "Macro sederhana 1", 64,
"Selamat datang"
```

- Kode selengkapnya:

```
Sub Main

msgbox "Macro sederhana 1", 64,
"Selamat datang"

End Sub
```

- Simpanlah macro dan dokumen dengan menekan shortcut CTRL-S.
- Tutuplah code editor. Kita akan segera kembali ke dokumen.

Untuk menjalankan macro yang baru saja kita buat, akseslah menu Tools -> Macros -> Run Macro...:

- Sebuah dialog Macro selector akan ditampilkan.
- Di panel kiri, pada Library, pilihlah macro1 -> Standard -> Module1.
- Di panel kanan, pada macro name, pilihlah Main.
- Klik tombol Run.
- Sebuah message box dengan judul Selamat Datang, dan pesan Macro sederhana 1 akan ditampilkan. Di dalam message box, sebuah icon information/tip dan sebuah tombol OK juga akan ditampilkan. Klik OK untuk menutup message box.

Berikut ini adalah penjelasan isi macro:

- Di macro tersebut, kita menampilkan se-

buah message box dengan pesan Macro sederhana 1, di mana macro tersebut dilengkapi dengan icon informasi/tip (kode: 64), sebuah tombol OK dan judul Selamat datang:

```
msgbox "Macro sederhana 1", 64, "Selamat datang"
```

Untuk tipe icon, kode-kode berikut ini bisa dipergunakan:

- 16: tanda Stop.
- 32: tanda tanya.
- 48: tanda seru.
- 64: tip.

Angka 64 tersebut kita tambahkan dengan 0 (tombol OK).

Untuk mengedit kembali macro kita, lakukanlah langkah-langkah berikut:

- Bukalah dokumen macro1.odt.
- Akseslah menu Tools -> Macros -> Organize Macros -> OpenOffice.org Basic...
- Sebuah dialog OpenOffice.org Basic Macros akan ditampilkan. Pada panel kiri, di bagian Macro from, pilihlah macro1.odt -> Standard -> Module1.
- Di panel kanan, pilihlah macro Main.
- Kliklah tombol Edit.

Macro sederhana 2

Di macro yang kedua ini, kita menyinggung sedikit macro tujuan, yaitu bagaimana mendapatkan nama file dari dokumen aktif. Hal ini diperlukan dalam macro tujuan karena kita akan menyimpan file PDF sesuai dengan nama dokumen.

Masih mempergunakan macro1.odt, tambahkan sebuah subbaru:

- Editlah module Module1. Bacalah contoh macro sederhana 1 untuk cara pengeditan macro.
- Tambahkan kode berikut ini di bagian paling bawah:

```
Sub GetActiveDocumentName
    url = StarDesktop.CurrentComponent
    msgbox url.URL,64, "Nama file aktif"
End Sub
```

Jalankan macro tersebut. Sebuah message box yang mengandung pesan nama file dokumen aktif akan ditampilkan, lengkap dengan title Nama file aktif dan sebuah icon tip.

Perlu dicatat bahwa nama file akan dikonversi ke format URL internet, RFC 1738, seperti contoh: *file:///home/nop/macro1.odt*. Untuk mendapatkan nama file UNIX, gunakanlah fungsi ConvertFromURL. Sebaliknya, agar file dikenali oleh OpenOffice.org, selalu gunakan fungsi ConvertToURL dari nama file yang didapatkan.

Macro sederhana 3

Di macro yang ketiga ini, kita akan menyempurnakan macro kedua, dengan mendapatkan nama file PDF dari nama file dokumen. Untuk nama dokumen macro1.odt, maka seharusnya nama file PDF yang dihasilkan adalah macro1.pdf, dan bukannya macro1.odt.pdf.

Masih mempergunakan macro1.odt, tambahkan sebuah subbaru:

- Editlah module Module1. Bacalah contoh macro sederhana 1 untuk cara pengeditan macro.
- Tambahkan kode berikut ini di bagian paling bawah:

```
Sub GetPDFFilenameFromActiveDocument
    url = StarDesktop.CurrentComponent
    pdf_file = Left( url.URL, Len( url.URL) - 4 ) + ".pdf"
    pdf_url = ConvertToURL (pdf_file)
    msgbox pdf_url,64, "Nama file PDF"
End Sub
```

Jalankan macro tersebut. Sebuah message box yang mengandung pesan nama file PDF dari dokumen aktif akan ditampilkan, lengkap dengan title Nama file PDF dan sebuah icon tip.

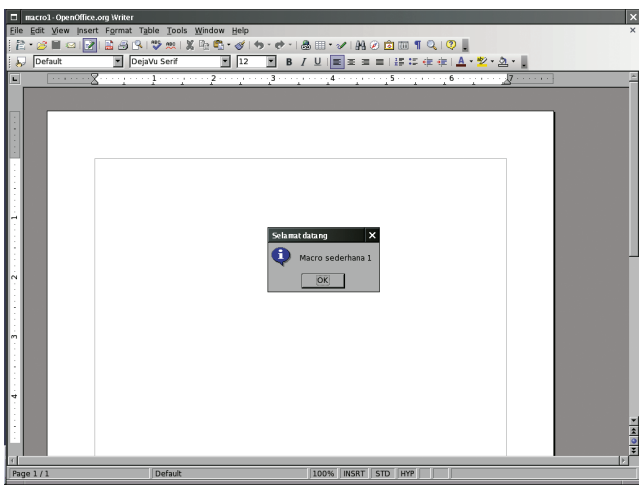
Menggunakan fungsi Left, kita akan mendapatkan nama file dokumen aktif dikurangi empat karakter terakhir dan kemudian menambahkan .pdf.

Keamanan

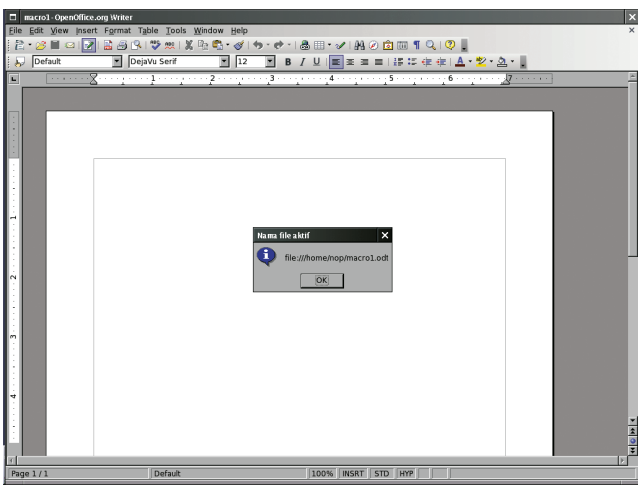
Tutuplah dokumen macro1.odt. Kemudian bukalah kembali. Sebuah dialog akan ditampilkan, di mana akan berisikan tiga tombol:

- Enable macros.
- Disable macros.
- Help.

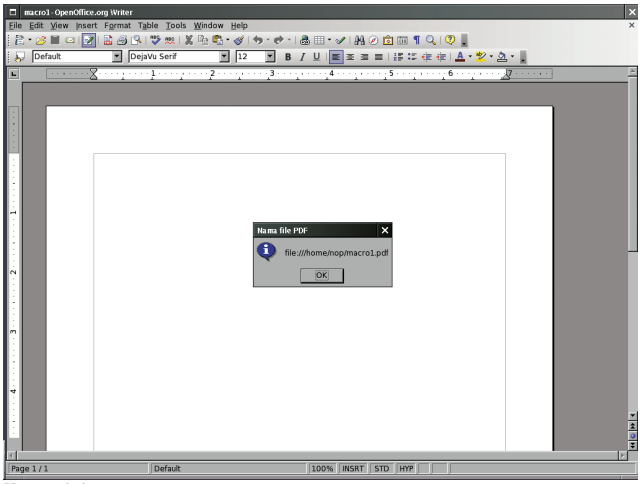
Untuk mengenable macro, kita bisa klik pada tombol *Enable Macros*. Sebaliknya, kita bisa klik pada tombol Disable Macros. Dengan cara demikian, OpenOffice.org berusaha menjaga keamanan penggunaan macro. User yang menerima sebuah file mengandung macro setidaknya telah diberitahu bahwa dokumen yang akan dibuka mengandung macro dan macro bisa mengandung virus.



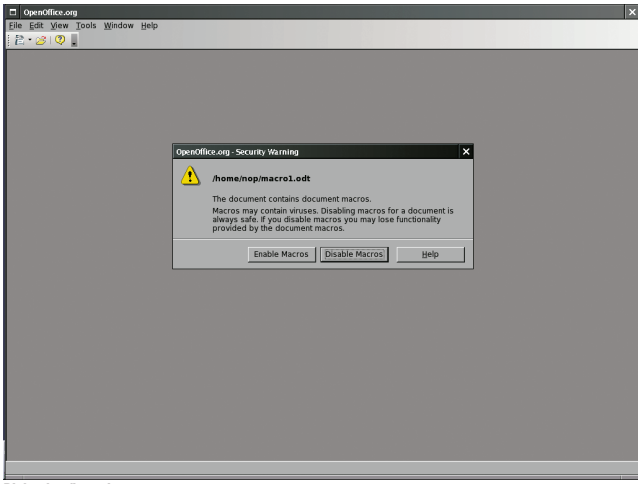
Macro sederhana 1.



Macro sederhana 2.



Macro sederhana 3.



Dialog konfirmasi macro.

Walau demikian, pengaturan keamanan ini masih bisa di-*customize* lagi, dengan mengakses menu Tools -> Options:

- Pada dialog yang ditampilkan, di panel kiri, pilihlah OpenOffice.org -> Security.
- Di panel kanan, kliklah tombol Macro Security
- Sebuah dialog akan ditampilkan.
- Terdapat empat level keamanan:
  - Very high.
  - High.
  - Medium.
  - Low.

Jangan sampai pengaturan diset ke low, di mana macro akan selalu dijalankan tanpa konfirmasi.

Ketika Anda terlibat dengan macro, pastikan keamanan menjadi pertimbangan utama. Dengan klik enable macros saja (pada dialog konfirmasi), virus dapat segera menyebar. Anda tidak harus menjalankan macro secara manual. Di macro penyimpanan otomatis file PDF yang kita buat, kita tidak pernah menjalankan macro tersebut secara manual, namun secara otomatis akan dijalankan.

Lokasi penyimpanan macro

Macro dapat disimpan pada dua tempat:

- Ikut sistem (komputer/user).
- Ikut dokumen.

Pada saat pemilihan macro yang ingin dijalankan atau diedit, kita selalu bisa memilih macro dari lokasi dokumen ataupun OpenOffice.org dan macro lokal.

Ketika macro disimpan ikut sistem, maka macro tersebut selalu tersedia untuk dokumen apapun. Ketika macro disimpan ikut dokumen, maka macro tersebut selalu tersedia di sistem manapun dokumen tersebut dibuka.

Event

Macro yang akan kita buat sepenuhnya mempergunakan bantuan event. OpenOffice.org mendukung eksekusi macro pada event-event tertentu yang telah ditentukan.

Event dapat dianalogikan sama seperti pada kehidupan sehari-hari. Contoh event adalah ketika kita merasa lapar, ketika hujan turun, dan lain sebagainya. Untuk event-event yang ada, kita bisa mengatur event handler tertentu. Di OpenOffice.org, event handler secara sederhana mempergunakan macro. Di kehidupan sehari-hari, contoh event handler untuk event lapar adalah makan.

Event-event yang telah didefinisikan bisa dilihat dengan langkah-langkah berikut:

- Akseslah menu Tools -> Customize.
- Sebuah dialog Customize akan ditampilkan.
- Aktiflah pada tab Events.

Untuk meng-*assign* event handler untuk event yang diinginkan, ketika masih berada pada daftar event, lakukanlah langkah-langkah berikut:

- Pilihlah event yang diinginkan.
- Klik tombol Macro...
- Sebuah dialog Macro Selector akan ditampilkan.
- Pilihlah macro yang ingin diassign, kemudian kliklah tombol OK.

Untuk menghilangkan *event handler* untuk event tertentu, ketika masih berada pada daftar event, lakukanlah langkah-langkah berikut:

- Pilihlah event yang diinginkan.
- Klik tombol Remove.

Assignment event handler juga bisa disimpan pada lokasi tertentu, seperti halnya macro. Perhatikanlah combo Save In, yang terletak di bawah daftar event. Agar event handler berlaku untuk semua dokumen, pilihlah Save In OpenOffice.org dan gunakan macro yang ikut sistem sebagai handler.

Sebagai catatan, event yang tersedia ketika disimpan ke dokumen dan ketika disimpan ikut sistem akan berbeda. Beberapa event hanya tersedia ketika ikut dokumen.

Otomatis menyimpan ke PDF

Kita memasuki pembahasan utama. Sebelum kita memulai, kita sepakati terlebih dahulu bahwa macro akan disimpan ikut sistem di My Macros -> Standard -> Module1. Hal ini dimaksudkan agar macro selalu tersedia apapun dokumen yang dibuka.

Nama fungsi yang akan dipergunakan adalah AutoSavePDF, yang disimpan pada module My Macros -> Standard -> Module1.

Berikut ini adalah isi fungsi AutoSavePDF:

```
Sub AutoSavePDF
   Dim Prop(0) as New com.sun.star.
beans.PropertyValue

   doc = StarDesktop.
CurrentComponent
   doc_URL = doc.URL
   doc_URL_dec = ConvertFromURL
(doc_URL)
```

```
    pdf_file = Left( doc_URL_dec, Len(
doc_URL_dec ) - 4 ) + ".pdf"
    pdf_url = ConvertToURL (pdf_file)

    Prop(0).Name = "FilterName"
    Prop(0).Value = "writer_pdf_
Export"

    doc.storeToUrl (pdf_url, Prop())

    msgbox "Done" + chr(13) + "File
" + pdf_file + " generated" , 64,
"Result"

End Sub
```

Dan, yang paling penting, assignlah macro ini untuk event Document has been saved, Save In OpenOffice.org. Setelah ini kita lakukan, setiap kali kita menyimpan dokumen sebuah file PDF akan dibuat secara otomatis dan setelah itu, sebuah pesan akan ditampilkan.

Penjelasan source code:

- Dapatkan terlebih dahulu nama dokumen aktif. Setelah itu, kita konversi ke nama file normal.

```
    doc = StarDesktop.
CurrentComponent
    doc_URL = doc.URL
    doc_URL_dec = ConvertFromURL
(doc_URL)
```

- Aturlah nama file PDF yang akan dibentuk:

```
    pdf_file = Left( doc_URL_dec,
Len( doc_URL_dec ) - 4 ) + ".pdf"
    pdf_url = ConvertToURL
(pdf_file)
```

- Properti penyimpanan kita berikan FilterName berupa writer_pdf_Export.

```
    Dim Prop(0) as New com.sun.
star.beans.PropertyValue

...
...
...
    Prop(0).Name = "FilterName"
    Prop(0).Value = "writer_pdf_
Export"
```

- Untuk menyimpan dokumen, gunakan perintah berikut. Perhatikan bahwa properti penyimpanan kita lewatkan pada storeToUrl().

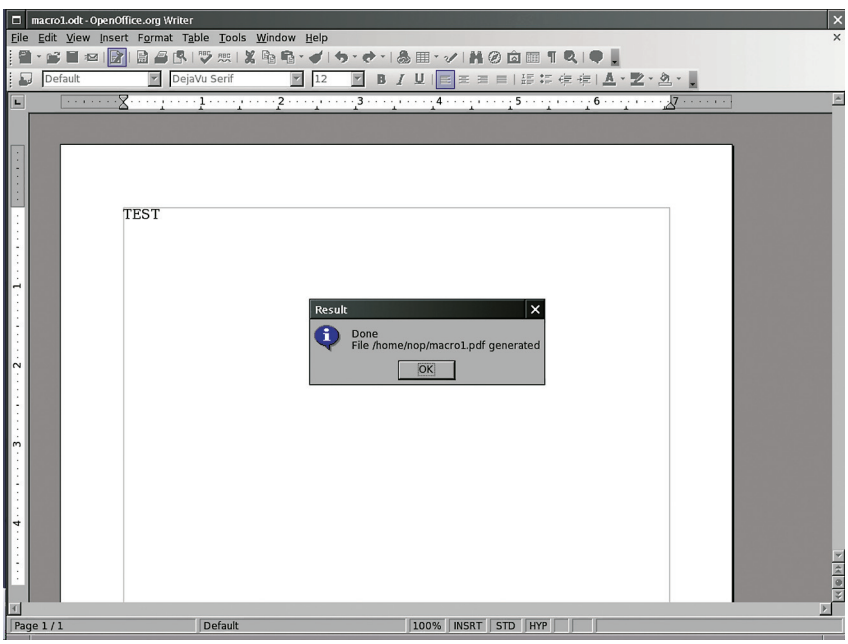
```
    doc.storeToUrl (pdf_url,
Prop())
```

- Terakhir, kita menampilkan message box:

```
    msgbox "Done" + chr(13) +
"File " + pdf_file + " generated"
, 64, "Result"
```

Bagaimana, seru *kan*? Sesuaikanlah dengan kebutuhan Anda. Sampai di sini dulu pembahasan kita. Di edisi-edisi berikutnya, kita akan membahas contoh-contoh macro OpenOffice.org lainnya.

Noprianto [noprianto@infolinux.co.id]



AutoSavePDF.

Penanganan Signal di Linux

Ketika suatu proses membandel, kita umumnya menggunakan program *kill* (atau program sejenis) untuk melakukan terminasi. Sederhananya, program kill hanya mengirimkan signal ke proses tersebut. Apa yang terjadi dan bagaimana menangani signal yang dikirimkan? Mari kita bahas di tulisan ini.

Di Linux, komunikasi proses umumnya dilakukan dengan bantuan signal. Proses satu mengirim signal ke proses lain dan proses yang dikirimkan signal tersebut akan merespon sesuai dengan kebutuhan.

Ketika suatu proses menerima signal, salah satu dari tiga hal berikut ini bisa dilakukan:

- Menerima apa adanya. Setiap signal memiliki aksi *default*. Dan, dalam hal ini, proses membiarkan aksi default itu terjadi. Contoh, aksi default proses ketika dikirimkan signal SIGTERM adalah terminasi proses. Maka, ketika proses menerima signal ini dan menerima apa adanya, proses akan diterminasi.
- Memblok signal. Signal yang diterima akan diabaikan. Terdapat dua signal yang tidak dapat diblok, yaitu SIGSTOP dan SIGKILL.
- Membelokkan atau menangkap signal. Alih-alih menerima apa adanya, proses akan menggunakan *signal handler* untuk melakukan aksi lain. Sebagai contoh, ketika proses dikirimkan signal SIGTERM, alih-alih proses diterminasi, program melakukan aksi lain, seperti *cleanup* misalnya. Terdapat dua signal yang tidak dapat dibelokkan/ditangkap, yaitu SIGSTOP dan SIGKILL.

Di Linux, program yang umum digunakan untuk mengirimkan signal adalah program kill. Walaupun namanya terdengar menyeramkan, tugas utama program ini adalah mengirimkan signal ke ID proses yang kita tentukan. Signal yang dikirimkan bisa kita pilih.

Program kill bukanlah program satu-satunya yang dapat digunakan untuk mengirimkan signal. Pengguna X umumnya menggunakan xkill. Pengguna KDE dan GNOME lain lagi. Program kita sendiri pun dapat memanggil system call kill() untuk mengirimkan signal ke Process ID tertentu. Kita pun pernah membahas ini di tulisan sebelumnya.

Berbeda dengan sistem operasi lain, penanganan signal di Linux sangatlah transparan. Di tulisan ini, kita akan membahas beberapa hal berikut:

- Mengatur alarm proses.
- Bekerja dengan signal set.
- Memblok signal.
- Membelokkan/menangkap signal.
- melakukan aksi tertentu berdasarkan signal yang telah disepakati.
- Membuat program pengunci terminal.

Semua contoh di tulisan ini dibangun di atas sistem Linux Slackware 11, Linux 2.6.18, namun seharusnya bisa diterapkan pada sistem lain tanpa masalah.

Sebelum melanjutkan, penulis menyarankan agar membaca juga pembahasan tentang *system call* untuk bekerja dengan proses di edisi sebelumnya, terutama untuk pembahasan contoh program kill.

Sekedar refresh singkat, berikut ini adalah prototype kill():

```
int kill(pid_t pid, int sig);
```

Argumen pertama adalah PID proses yang ingin dikirimkan signal dan argumen kedua adalah signal yang ingin dikirimkan. Daftar signal selengkapnya bisa dibaca di /usr/include/bits/signum.h atau bisa juga mempergunakan program kill, yang dijalankan dengan argumen -l.

Mengatur alarm proses

Masih berhubungan dengan signal, program dapat pula mengatur satu alarm internal, yang akan memicu pengiriman SIGALRM. Untuk mengirimkan alarm, kita bisa mempergunakan system call alarm():

```
#include <unistd.h>

unsigned int alarm(unsigned int seconds);
```

Contoh pertama penggunaan alarm() kita bahas di s_alarm.c:

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

int main(void)
{
  fprintf (stdout, "Set alarm for 5 seconds\n");
  alarm (5);

  while (1)
  {
   fprintf (stdout, "Idle...\n");
   sleep (1);
  }
```

```
  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_alarm s_alarm.c
```

Contoh output:

```
$ ./s_alarm
Set alarm for 5 seconds
Idle...
Idle...
Idle...
Idle...
Idle...
Alarm clock
```

Penjelasan:

- Kita mengatur alarm selama 5 detik, dan setelah sampai waktunya, SIGALRM akan dikirimkan dan memicu proses diterminasi.

Alarm juga bisa dibatalkan dengan memanggil alarm() dengan argumen 0, seperti contoh s_alarm2.c berikut:

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

int main(void)
{
  long int random_number;

  fprintf (stdout, "Set alarm for 1
second\n");
  fprintf (stdout, "Looking for odd
number\n");

  alarm (1);

  srandom (time(0) * getpid());

  while (1)
  {

   random_number = random();

   fprintf (stdout, " %ld ", random_
number);

   if (random_number % 2 == 1)
   {
    fprintf (stdout, "\nFound number
%ld\n", random_number);
    fprintf (stdout, "\nAlarm
cancelled.\n");
    alarm (0);
    return 0;
   }
  }

  return 1;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_alarm2 s_alarm2.c
```

Contoh output:

```
$ ./s_alarm2
Set alarm for 1 second
Looking for odd number
 1627760320  1261929164  1186525202
1238754019
Found number 1238754019

Alarm cancelled.
```

Penjelasan:

- Kita mengatur alarm selama 1 detik.
- Setelah itu, kita mulai mendapatkan nomor acak.
- Apabila nomor acak ganjil ditemukan, maka alarm dibatalkan.
- Apabila nomor acak ganjil tidak juga didapatkan selama 1 detik, maka alarm akan dikirimkan dan memicu proses diterminasi.

Bekerja dengan signal set

Kita akan memasuki pembahasan tentang penanganan signal. Sebagai langkah pertama, kita akan berkenalan dengan signal set, yang memungkinkan kita untuk mendefinisikan sekumpulan signal yang nantinya bisa kita tangani.

Terdapat beberapa cara membuat signal set:

- Mendaftarkan set kosong (empty set) dan mulai menambahkan signal satu demi satu. Sangat berguna kalau Anda ingin menangani satu atau dua signal.
- Mendaftarkan set penuh dan mulai mengurangi satu demi satu. Sangat berguna kalau Anda ingin menangani banyak signal.

Beberapa fungsi yang bisa dipergunakan:

```
#include <signal.h>

int sigemptyset(sigset_t *set);

int sigfillset(sigset_t *set);

int sigaddset(sigset_t *set, int
signum);

int sigdelset(sigset_t *set, int
signum);

int sigismember(const sigset_t
*set, int signum);
```

- sigemptyset() dipergunakan untuk membuat set kosong.
- sigfillset() digunakan untuk membuat set penuh.
- sigaddset() digunakan untuk menambahkan signal.
- sigdelset() digunakan untuk mengurangi signal.
- sigismember() digunakan untuk memeriksa apakah suatu signal menjadi anggota set.

Untuk lebih jelasnya, di beberapa contoh berikut, kita akan mengamati apakah SIGTERM dan SIGINT termasuk dalam set.

Kita akan mulai dengan contoh pertama, s_signal_empty.c:

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

int main(void)
{
  sigset_t sigset;

  sigemptyset (&sigset);

  if (sigismember (&sigset,
SIGTERM))
  {
   fprintf (stdout, "SIGTERM in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGTERM NOT in
signal mask\n");
  };


  if (sigismember (&sigset, SIGINT))
  {
   fprintf (stdout, "SIGINT in
signal mask\n");
  }
  else
  {
```

```
   fprintf (stdout, "SIGINT NOT in
signal mask\n");
  };


  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_empty s_signal_
empty.c
```

Contoh output:

```
$ ./s_signal_empty
SIGTERM NOT in signal mask
SIGINT NOT in signal mask
```

Penjelasan:

- Kita membuat set empty dan oleh karenanya, SIGTERM dan SIGINT tidak termasuk dalam set, karena kita tidak menambahkan mereka.

Contoh kedua kita sajikan dalam s_signal_add.c:

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

int main(void)
{
  sigset_t sigset;

  sigemptyset (&sigset);

  sigaddset (&sigset, SIGINT);

  if (sigismember (&sigset,
SIGTERM))
  {
   fprintf (stdout, "SIGTERM in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGTERM NOT in
signal mask\n");
  };


  if (sigismember (&sigset, SIGINT))
  {
   fprintf (stdout, "SIGINT in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGINT NOT in
signal mask\n");
  };


  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_add s_signal_add.c
```

Contoh output:

```
$ ./s_signal_add
SIGTERM NOT in signal mask
SIGINT in signal mask
```

Penjelasan:

- Kita mulai dengan set kosong dan menambahkan SIGINT, dengan sigaddset(). Oleh karenanya, SIGINT masuk dalam anggota.

Contoh ketiga kita sajikan dalam s_signal_fill.c:

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

int main(void)
{
  sigset_t sigset;

  sigfillset (&sigset);

  if (sigismember (&sigset,
SIGTERM))
  {
   fprintf (stdout, "SIGTERM in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGTERM NOT in
signal mask\n");
  };


  if (sigismember (&sigset, SIGINT))
  {
   fprintf (stdout, "SIGINT in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGINT NOT in
signal mask\n");
  };


  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_fill s_signal_fill.c
```

Contoh output:

```
$ ./s_signal_fill
SIGTERM in signal mask
SIGINT in signal mask
```

Penjelasan:

- Kita mulai dengan set penuh dan oleh karenanya SIGTERM dan SIGINT masuk dalam anggota.

Contoh terakhir adalah s_signal_del.c:

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

int main(void)
{
  sigset_t sigset;

  sigfillset (&sigset);

  sigdelset (&sigset, SIGINT);

  if (sigismember (&sigset,
SIGTERM))
  {
   fprintf (stdout, "SIGTERM in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGTERM NOT in
signal mask\n");
  };


  if (sigismember (&sigset, SIGINT))
  {
   fprintf (stdout, "SIGINT in
signal mask\n");
  }
  else
  {

   fprintf (stdout, "SIGINT NOT in
signal mask\n");
```

```
  };


  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_del s_signal_del.c
```

Contoh output:

```
$ ./s_signal_del
SIGTERM in signal mask
SIGINT NOT in signal mask
```

Penjelasan:

- Kita mulai dengan set penuh dan mengurangi SIGINT dengan sigdelset(). Oleh karenanya, SIGINT tidak masuk dalam anggota.

Bosan melihat contoh-contoh ini? Bingung dengan penerapannya, lihatlah pembahasan berikut.

Memblok signal

Di bagian ini, kita akan membahas bagaimana memblok signal yang bisa dikirimkan. Sekedar mengingatkan, SIGKILL dan SIGSTOP tidak bisa diblok.

Kita akan mempergunakan signal set yang kita bahas sebelumnya. Sebagai contoh, kalau kita ingin memblok semua signal, maka kita bisa mempergunakan set penuh. Kalau kita ingin memblok satu signal, kita bisa memulai dengan set kosong dan menambahkan signal yang diinginkan. Bisa diduga, contoh-contoh sebelumnya akan terasa berguna.

Contoh berikut ini, s_signal_block.c akan memperlihatkan kita memblok SIGINT (Signal interrupt). SIGINT umumnya dikirimkan ketika kita menekan kombinasi tombol CTRL-C di dalam program.

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

int main(void)
{
  sigset_t sigset;

  pid_t pid;

  pid = getpid();

  sigemptyset (&sigset);

  sigaddset (&sigset, SIGINT);

  sigprocmask (SIG_BLOCK, &sigset,
NULL);

  while (1)
  {
   fprintf (stdout,"I am %d. Try
interrupt me...\n", pid);
  }

  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_block s_signal_
block.c
```

Contoh output:

```
$ ./s_signal_block
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
I am 3056. Try interrupt me...
```

Cobalah untuk menekan CTRL-C. Pastinya, program tidak diterminasi karena SIGINT sudah kita blok :). Karena kita tidak memblok signal lain, Anda bisa menterminasi proses dengan menekan tombol CTRL-\ (mengirimkan SIGQUIT) atau mengirimkan signal secara eksplisit dengan program kill.

Penjelasan:

- Dengan sigprocmask(), kita memeriksa dan mengubah signal terblok:

```
        #include <signal.h>

        int sigprocmask(int how,
  const sigset_t *set, sigset_t
  *oldset);
```

Membelokkan/menangkap signal

Kita akan masuk ke pembahasan terakhir, bagaimana kita membelokkan atau menangkap signal yang dikirimkan. Perlu diingatkan sekali lagi, kita tidak bisa membelokkan atau menangkap SIGKILL dan SIGSTOP.

Beberapa catatan penting:

- Kita perlu mendefinisikan signal handler yang merupakan fungsi yang tidak mengembalikan nilai (void) dan menerima satu argumen bertipe int.
- Kita menggunakan struktur data sigaction untuk mengatur handler dan properti lainnya dalam penanganan signal.
- Selanjutnya, dengan fungsi sigaction(), kita mendaftarkan signal yang ingin ditangani.

```
        #include <signal.h>


                struct sigaction {
                    void (*sa_
  handler)(int);
                    void (*sa_
  sigaction)(int, siginfo_t *, void
  *);
                    sigset_t
  sa_mask;
                    int sa_flags;
                    void (*sa_
  restorer)(void);
                }


        int sigaction(int signum,
  const struct sigaction *act,
  struct sigaction
        *oldact);
```

Kita akan masuk langsung ke contoh pertama, s_signal_catch.c, yang akan membelokkan SIGINT, SIGQUIT dan SIGTERM ke fungsi tertentu. Program akan menunggu 10 detik untuk menerima signal.

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

void signal_handler (int signal)
{
  fprintf (stdout, "\nCaught signal
number %d\n", signal);

}

int main(void)
{
  struct sigaction signal_action;

  pid_t pid;

  pid = getpid();

  signal_action.sa_handler = signal_
handler;
  signal_action.sa_flags = SA_
NODEFER;

  sigaction (SIGINT, &signal_action,
```

```
NULL);
  sigaction (SIGTERM, &signal_
action, NULL);
  sigaction (SIGQUIT, &signal_
action, NULL);

  fprintf (stdout,"I am %d. Sleeping
for 10 secs. Try send me some
signals...\n", pid);

  sleep (10);

  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_catch s_signal_
catch.c
```

Contoh output:

```
$ ./s_signal_catch
I am 3136. Sleeping for 10 secs. Try
send me some signals...
[TEKAN CTRL-C]
Caught signal number 2
```

Ketika program berjalan, cobalah untuk menekan CTRL-C, yang akan mengirimkan SIGINT, signal nomor 2. Program sudah mengatur agar SIGINT akan ditangani oleh signal_handler().

Pertanyaan penting: kalau sudah kita tangani, kenapa program diterminasi juga? Secara prinsip, perlu kita pahami bahwa SIGINT sudah kita tangani. Bahwa program diterminasi, karena memang sudah tidak ada kode-kode yang dikerjakan lagi.

Di contoh kedua, s_signal_catch2.c, kita akan melihat contoh yang lebih jelas, dimana program akan menangani SIGINT, SIGTERM dan SIGQUIT, namun hanya akan keluar apabila SIGQUIT dikirimkan. Kalau dikirimkan SIGINT atau SIGTERM, program tidak akan peduli.

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

int quit = 0;

void signal_handler (int signal)
{
  fprintf (stdout, "\nCaught signal
number %d\n", signal);

  if (signal == 3)
  {
   quit = 1;
   fprintf (stdout, "Will quit.\n");
  }
  else
  {
   quit = 0 ;
   fprintf (stdout, "I don't care.\
n");
  }
}

int main(void)
{
  struct sigaction signal_action;

  pid_t pid;

  pid = getpid();

  signal_action.sa_handler = signal_
handler;
  signal_action.sa_flags = SA_
NODEFER;

  sigaction (SIGINT, &signal_action,
NULL);
  sigaction (SIGTERM, &signal_
action, NULL);
  sigaction (SIGQUIT, &signal_
action, NULL);

  fprintf (stdout,"I am %d. Try send
me some signals...\n", pid);

  while (quit == 0)
  {

  }
  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_catch2 s_signal_
catch2.c
```

Contoh output:

```
$ ./s_signal_catch2
I am 3167. Try send me some
signals...
[TEKAN CTRL-C]
Caught signal number 2
I don't care.
[TEKAN CTRL-C]
Caught signal number 2
I don't care.
[TEKAN CTRL-\]
Caught signal number 3
Will quit.
```

Bisa kita lihat, kita menangani ketiga-tiganya: SIGINT, SIGQUIT dan SIGTERM, namun hanya SIGQUIT yang mengakibatkan program diterminasi. Kita menggunakan variabel global quit untuk membantu kita. Program akan terus mengulang dan hanya akan keluar apabila quit bernilai 1.

Di contoh ketiga, s_signal_catch3.c, program menjadi lebih terstruktur:

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

int quit = 0;

void signal_handler_int(void)
{
  fprintf (stdout, "I am handling
SIGINT\n");
  fprintf (stdout, "I don't care.\
n");
  quit = 0;
}

void signal_handler_term(void)
{
  fprintf (stdout, "I am handling
SIGTERM\n");
  fprintf (stdout, "I don't care.\
n");
  quit = 0;
}

void signal_handler_quit(void)
{
  fprintf (stdout, "I am handling
SIGQUIT\n");
  fprintf (stdout, "Will quit.\n");
  quit = 1;

}

void signal_handler (int signal)
{
  switch (signal)
  {
   case 2: signal_handler_int();
     break;
   case 3: signal_handler_quit();
     break;
   case 15: signal_handler_term();
```

```
    break;

  }

}

int main(void)
{
  struct sigaction signal_action;

  pid_t pid;
  pid = getpid();

  signal_action.sa_handler = signal_
handler;
  signal_action.sa_flags = SA_
NODEFER;

  sigaction (SIGINT, &signal_action,
NULL);
  sigaction (SIGTERM, &signal_
action, NULL);
  sigaction (SIGQUIT, &signal_
action, NULL);

  fprintf (stdout,"I am %d. Try send
me some signals...\n", pid);

  while (quit == 0)
  {

  }
  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_signal_catch3 s_signal_
catch3.c
```

Contoh output:

```
$ ./s_signal_catch3
I am 3207. Try send me some
signals...
[TEKAN CTRL-C]
I am handling SIGINT
I don't care.
[TEKAN CTRL-C]
I am handling SIGINT
I don't care.
[TEKAN CTRL-\]
I am handling SIGQUIT
Will quit.
```

Contoh kasus1: acak ketika menerima SIGUSR1

SIGUSR1 umumnya digunakan untuk signal custom yang dibutuhkan oleh program. Di contoh berikut ini, ketika program menerima SIGUSR1, nomor acak akan didapatkan dan ditampilkan.

Source code s_random.c:

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

void print_random_number(void)
{
  int number;

  number = rand();

  fprintf (stdout, "\nRandom number:
%d", number);
}

void signal_handler (int signal)
{
  print_random_number();
}

int main(void)
{
  int pid;

  struct sigaction signal_action;

  pid = getpid();

  signal_action.sa_handler = signal_
handler;

signal_action.sa_flags = 0;

  sigaction (SIGUSR1, &signal_
action, NULL);

  fprintf (stdout, "I am process
with ID: %d. Waiting for signal
USR1...\n\n", pid);

  while (1)
  {
  }
  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_random s_random.c
```

Contoh output:

```
$ ./s_random
I am process with ID: 3220. Waiting
for signal USR1...

Random number: 1804289383
Random number: 846930886
Random number: 1681692777
...
...
```

Gunakan program kill untuk mengirimkan SIGUSR1:

```
$ kill -USR1 3220
```

Contoh kasus 2: screen locker

Contoh program berikut ini, screen locker, akan memblok semua signal yang bisa diblok (kecuali SIGKILL dan SIGSTOP.

Source code s_screenlock.c:

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

int main(void)
{
  sigset_t sigset;

  pid_t pid;

  pid = getpid();

  sigfillset (&sigset);

  sigprocmask (SIG_BLOCK, &sigset,
NULL);

  fprintf (stdout,"I am screen
locker with ID: %d\n", pid);

  while (1);
  return 0;
}
```

Lakukan kompilasi dengan perintah berikut:

```
$ gcc -o s_screenlock s_screenlock.c
```

Contoh output:

```
$ ./s_screenlock
I am screen locker with ID: 3244
```

Untuk melakukan terminasi, kirimkan signal SIGKILL.

Sampai di sini dulu pembahasan kita. Bacalah manual setiap system call untuk penjelasan selengkapnya. Selamat menerapkan ke dalam aplikasi Anda, apabila dibutuhkan.

Noprianto [noprianto@infolinux.co.id]

Penjelasan Pesan Kesalahan Umum di Linux

Stabil atau tidaknya suatu sistem operasi, tidak terlepas dari kemampuan kernel untuk menangani proses. Linux sendiri termasuk salah satu sistem operasi yang cukup hebat menangani proses. Di tulisan ini, kita akan membahas beberapa *system call* untuk bekerja dengan proses di Linux.

Sebagai seorang pemula, Anda tentu kadang merasa bingung dengan beberapa pesan kesalahan di shell command line Linux yang cukup singkat tanpa penjelasan. Misalnya, saat ingin menjalankan sebuah perintah, hasilnya "Permission denied". Tapi kenapa tepatnya permission denied, padahal Anda sudah login sebagai root, yang mana seharusnya sudah bisa melakukan apapun? Atau "command not found". Kok bisa? Padahal nama perintahnya sudah benar sesuai yang ada di buku/manual, tidak salah ketik.

Artikel ini menjelaskan beberapa pesan kesalahan umum secara lebih mendetil: maksud pesan kesalahan, beberapa penyebab umum munculnya pesan kesalahan tersebut, disertai tip-tip yang relevan.

Command not found

Pesan kesalahan ini muncul jika shell tidak bisa menemukan perintah yang Anda inginkan.

Salah satu penyebab yang paling umum bagi pemula, memang, adalah salah ketik atau lupa apa nama persis perintahnya. Nama-nama perintah di Linux kadang "kriptik" atau aneh, sehingga sulit diingat. Sebagian nama perintah amat disingkat-singkat berupa 2-3 huruf saja, karena dulu memang di Unix ruang memori dan disk amat amat terbatas sehingga semua serba superpendek. Contohlah: ls, pwd, du, df. Jangankan pemula, pengguna veteran pun kadang lupa dengan berbagai perintah atau skrip yang jumlahnya ribuan itu. Katakanlah a2ensite atau a2dismod, dua skrip helper di Debian yang sering sekali saya lupa sampai sekarang. Belum lagi kadang tiap distro memiliki nama perintah yang berbeda: di distro yang satu "useradd", di yang lain "adduser".

Penyebab lain adalah memang program yang Anda inginkan belum terinstal. Saat Anda menginstal sebuah sistem baru, kadang setting default dari distro adalah minimalis. Sebuah kebijakan yang cukup baik sebetulnya. Contohnya, saat menginstal sebuah sistem Debian tanpa GUI dan berbagai embel-embel, bahkan perintah seperti less pun belum ada. Apalagi utilitas popular seperti mc atau wget atau editor favorit Anda, yang harus diinstal dulu. Cara menginstalnya tentu di luar bahasan artikel ini, karena berbeda-beda sesuai distro (apt-get, yum, klik, dll) dan selera (GUI, TUI, atau command line).

Penyebab lain yang umum adalah, programnya ada tapi berada di lokasi yang tidak dicari oleh shell. Perlu dicatat bahwa saat Anda mengetikkan nama sebuah perintah di shell seperti ini:

```
$ ketuk
```

Maka shell seperti bash biasanya mencari ketuk ini di daftar fungsi atau alias yang telah didefinisikan sebelumnya, atau program atau skrip bernama ketuk yang ada di filesystem. Untuk program atau skrip yang ada di filesystem, shell akan mencari hanya di lokasi tertentu saja yaitu sesuai yang didefinisikan di variabel lingkungan PATH. Mari kita lihat isi PATH:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/bin/
X11:/bin
```

Artinya adalah, shell hanya akan mencari di direktori /usr/local/bin, /usr/bin, /usr/bin/X11, dan /bin, dalam urutan tersebut. Program-program yang berada di lokasi yang lain, seperti /sbin, /usr/sbin, /opt/VirtualBox-1.3.8, bahkan di direktori yang saat ini Anda berada (current directory) pun akan diabaikan. Jadi Anda mungkin bingung, kok program yang berada di depan mata, yang jika di-"ls" sudah jelas-jelas ada, ketika ingin dieksekusi kok tetap saja dianggap "not found"?

Untuk mengeksekusi program-program di luar lokasi standar yang ada di PATH, Anda dapat menyebutkan lokasinya sebelum nama program, misalnya:

```
$ /opt/Ketuk-1.2/bin/ketuk
```

Atau jika Anda saat ini sedang berada di direktori /opt/Ketuk-1.2/doc, Anda bisa menggunakan lokasi relatif:

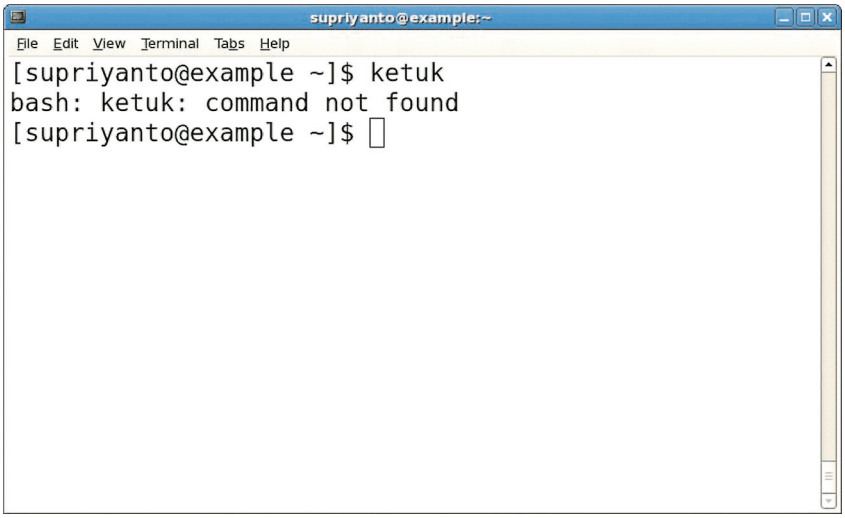
```
$ ../bin/ketuk
```

Salah satu kasus yang umum terjadi adalah: di beberapa distro jika Anda berubah menjadi root (menggunakan perintah "su"), maka PATH tetap belum berubah. Akibatnya, jika Anda mencoba menjalankan perintah:

```
# traceroute www.google.com
```

maka shell tidak menemukan program traceroute ini, yang mana umumnya berada di lokasi /usr/sbin, yang tidak terdaftar dalam PATH standar user normal. Untuk mem-

IKLAN



Pesan kesalahan karena perintah tidak terdapat di Linux.

peroleh PATH milik root yang mengandung lokasi-lokasi sbin: /sbin, /usr/sbin, /usr/local/sbin maka saat berganti menjadi root Anda perlu menggunakan perintah "su -".

Ngomong-ngomong, mengapa kelakuan shell seperti itu, hanya mencari program di PATH? Yang pertama, jelas, supaya proses pencarian program tidak memakan waktu terlalu panjang. Di dalam sebuah sistem bisa ada ratusan hingga ribuan direktori dan tidak ada indeks yang sifatnya real time, terlalu memakan waktu jika shell mencoba menelusuri semua direktori ini satu per satu.

Alasan lain adalah, dalam kasus current directory, demi keamanan. Program hanya akan dicari di lokasi yang sudah tertentu dan diketahui aman (/usr/bin, /bin, dll). Current directory kadang "tidak aman". Anda bisa saja sedang berjalan-jalan ke /tmp atau direktori milik user lain. Bisa aja ada nama program bernama "ls", "cat", "less" di direktori tersebut tapi isinya sama sekali lain. Sekarang bayangkan Anda seorang admin atau staf shared hosting, dan salah satu user Anda berpura-pura meminta bantuan Anda untuk melihat sebuah direktori miliknya, dengan tujuan agar Anda "tak sengaja" menjalankan program "ls", syukur-syukur sebagai root. Program ini bisa saja bertujuan untuk menginstal malware atau memasang backdoor bagi si user. Nah, sudah mengerti bukan mengapa secara defaultnya shell tidak memasukkan current directory dalam PATH?

Sebagai catatan, untuk mengeksekusi sebuah perintah yang berada di current directory, gunakan sintaks:

```
$ ./ketuk
```

Dan jika Anda tetap ngotot ingin agar dapat mengeksekusi program-program yang ada current directory secara polos, maka Anda dapat menambahkan current directory ini (".") ke dalam PATH, contoh:

```
$ export PATH="$PATH:."
```

Perintah tadi dapat ditambahkan misalnya ke ~/.bash_profile Anda agar dieksekusi setiap kali Anda login. Perlu diingat bahwa menambahkan current directory ke PATH amat tidak dianjurkan karena risiko keamanannya, terutama jika ini dilakukan untuk user root.

Permission denied

Pesan kesalahan ini muncul jika Anda dianggap tidak memiliki hak yang cukup untuk menjalankan program.

Beberapa program, terutama yang sifatnya setuid, diproteksi dari user atau group tertentu. Misalnya:

```
$ ls -l /bin/su
-rwsr-x--- 1 root su-user     27000
2006-12-08 01:28 su
```

Di sini terlihat bahwa hanya root dan anggota group "su-user" saja yang diizinkan mengakses /bin/su.

Kadang-kadang Anda mengalami program yang baru Anda kopi atau upload tidak bisa dieksekusi dengan pesan error permission denied ini. Umumnya ini karena permission file belum ada bit x (execute), contoh 0644 dan belum 0755:

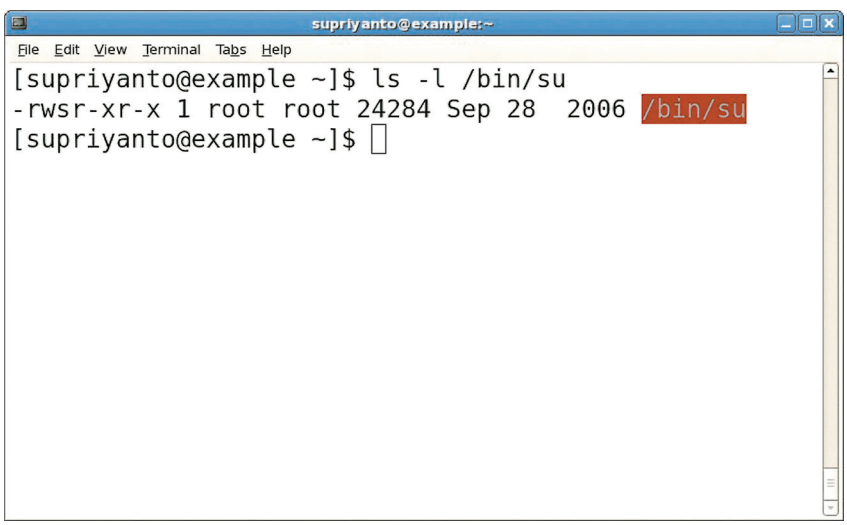
```
$ ls -l ketuk
-rw-r--r-- 1 steven steven 373
2007-05-05 06:42 ketuk
```

Untuk menjalankan sebuah program, dibutuhkan bit x (execute) pada permission. Meskipun Anda root, kernel tetap akan menolak mengeksekusi sebuah program atau skrip yang tidak memiliki bit x.

Untuk memperbaiki masalah ini, berikan bit x pada program:

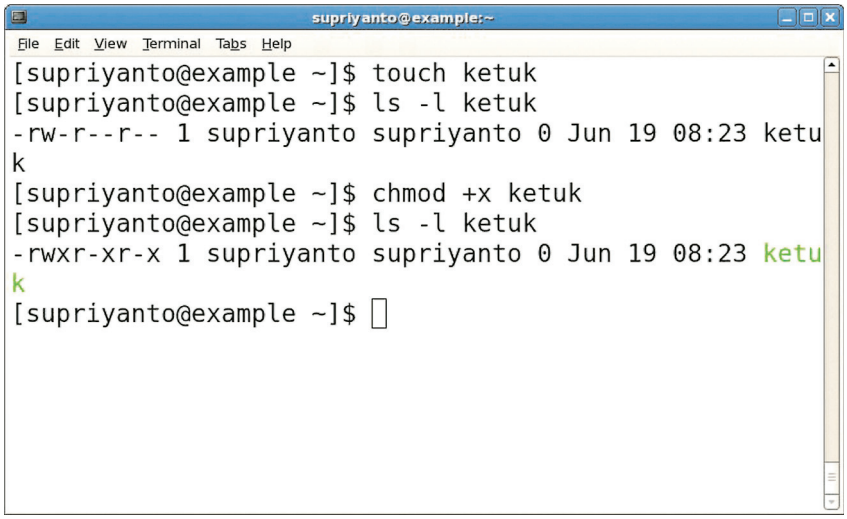
```
$ chmod +x ketuk
$ ls -l ketuk
-rwxr-xr-x 1 steven steven 373
2007-05-05 06:42 ketuk
```

Catatan: jika programnya adalah sebuah skrip, maka Anda bisa juga memanggil skrip ini dengan diawali nama interpreter/bahasa pemrogramannya. Dengan cara ini, Anda tidak membutuhkan bit x pada si skrip melainkan cukup bit r (read) saja. Contoh, jika ketuk pada contoh di atas adalah skrip Perl:



Melihat permission dari perintah su.

```
$ perl ketuk
```



Ubah permission suatu file menjadi executable.

Connection refused, Connection timed out, Connection reset by peer

Tiga pesan kesalahan ini juga sering dijumpai dalam bekerja di jaringan, dan seorang pemula kadang tidak mengerti jelas maksudnya dan bedanya satu dengan yang lain. Penyebab munculnya pesan-pesan kesalahan ini memang beragam.

Jika diterjemahkan, "connection refused" artinya permintaan koneksi kita ditolak. Umumnya ini terjadi jika kita ingin menghubungi sebuah service yang sedang mati. Maksudnya mati di sini adalah tidak ada program (daemon, server) yang sedang mendengarkan (listen) di port yang ingin kita hubungi. Contoh, cobalah menghubungi sebuah port acak di komputer Anda sendiri:

```
$ telnet localhost 23056
Trying 127.0.0.1...
telnet: Unable to connect to remote
host: Connection refused
```

Ini memang karena umumnya tidak ada daemon yang listen pada port tersebut. Berbeda misalnya dengan port umum seperti 80 (HTTP). Jika sistem Anda sudah diinstal web server:

```
$ telnet localhost 80
Connected to localhost.
Escape character is '^]'.
```

Ini artinya kita berhasil konek ke port 80 untuk berbicara dengan web server. Tekan Ctrl-] (tahan tombol Control diikuti tekan tombol "]") untuk keluar.

Apa jadinya kalau webserver kita matikan? Misalnya:

```
# /etc/init.d/apache2 stop
(di distro tertentu mungkin Anda
menggunakan sudo /etc/init.d/
apache2 stop).
$ telnet localhost 80
Trying 127.0.0.1...
telnet: Unable to connect to remote
host: Connection refused
```

Kini hasilnya adalah Connection refused, karena tidak ada lagi program yang mendengarkan koneksi di port 80. Jadi kadang-kadang pesan kesalahan ini dapat dipakai untuk tujuan monitoring, karena menjadi indikasi apakah sebuah service hidup atau mati.

Tapi bisa saja pesan Connection refused terjadi karena firewall. Misalnya, di firewall ada rule untuk me-REJECT paket remote ke port 25, untuk mencegah program di komputer lokal menghubungi SMTP server remote (dan melakukan spam, misalnya). Maka jika kita mencoba:

```
$ telnet mx.yahoo.com 25
Trying 68.142.195.60...
telnet: Unable to connect to remote
host: Connection refused
```

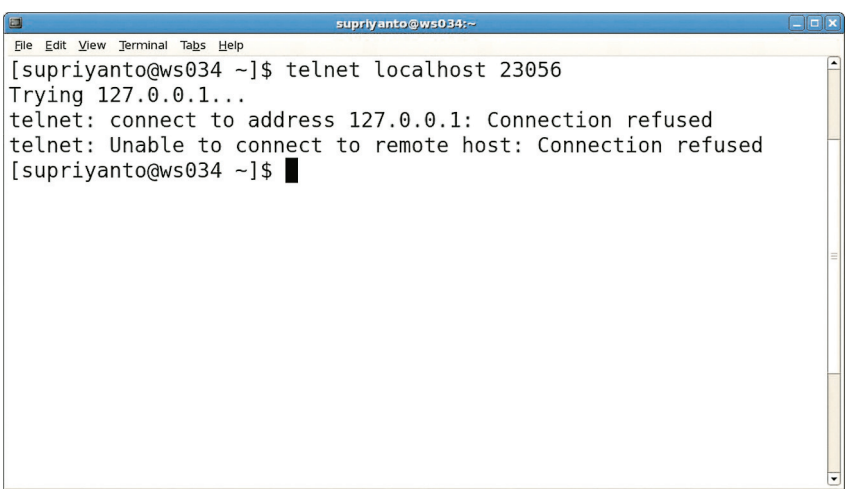
Belum tentu berarti service SMTP di Yahoo! mati, tapi mungkin saja ada firewall yang membuat koneksi Anda gagal.

"Connection timed out" artinya saat Anda mencoba menghubungi sebuah port remote, tidak ada jawaban balik dari ujung sana. Penyebabnya dapat berbeda-beda: bisa saja koneksi Anda ke remote host memang amat lambat sehingga limit waktu koneksi yang ditetapkan terlampaui. Bisa juga terdapat firewall yang memblokir dengan men-DROP (bukan me-REJECT) paket-paket yang dikirimkan sehingga permintaan koneksi Anda tidaklah pernah sampai ke tujuan.

Bagaimana dengan "Connection reset by peer"? Pesan kesalahan ini terjadi jika koneksi dengan remote host sudah terjadi, namun putus. Penyebab putusnya bisa antara lain: program di remote host sengaja memutuskan koneksi. Ini pun penyebabnya bisa macam-macam: IP Anda tidak disukai oleh si program, sudah ada terlalu banyak koneksi dari IP atau blok IP Anda, program di sisi sana tewas karena satu dan lain hal, dll. Penyebab lain adalah masalah koneksi internet, misalnya jika Anda menggunakan dialup dan saluran telepon terputus atau hang up.

Singkatnya, jika Anda mengalami problem koneksi jaringan, perlu melihat dulu apakah penyebabnya di sisi sistem Anda, di sisi sistem remote, atau di antara keduanya (ISP, firewall gateway, proxy, dll).

Steven Haryanto [steven@masterwebnet.com]



Contoh pesan kesalahan telnet.

Instal Zimbra dan Samba Menyerupai Exchange dan NT

Server Exchange dan PDC sudah menjadi "bagian hidup" bagi banyak perusahaan di Indonesia. Penulis yang pemula di dalam Linux, mencoba alternatif lain untuk menghormati hak cipta dan menekan biaya lisensi untuk perusahaan dengan menggunakan Zimbra dan Samba.

Pengguna server Microsoft Windows NT sebagai PDC (*Primary Domain Controller*) dan server Microsoft Exchange sebagai server mail pasti sangat senang dengan fasilitas *single login* yang ditawarkan. Pembuatan dan *maintenance userID* dan *password* dikelola oleh server PDC. Server Exchange hanya mengelola *mailbox*. Untuk menekan biaya lisensi, marilah kita alihkan tugas tersebut ke *open source* Zimbra dan Samba di dalam Linux.

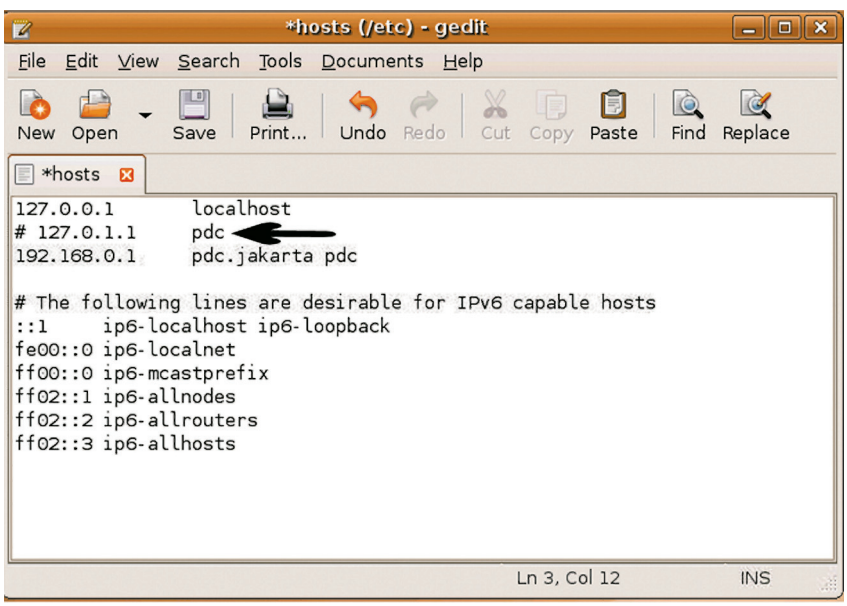
Dalam proses transisi menuju era open source, penulis berasumsi bahwa peralihan *back office* (salah satunya server) dilakukan paling awal. Untuk *client/workstation*, dilakukan bersamaan dengan perubahan kebiasaan pengguna.

Untuk penerapan di kantor, penulis menyiapkan satu komputer baru dengan instalasi baru Ubuntu 7.04.04. Pemilihan Ubuntu 7.04.04 ini disebabkan oleh database drivernya yang lebih komplit dibanding versi sebelumnya. Penulis memilih distro Ubuntu karena proses instalasinya sangat sederhana, cocok dengan penulis yang masih pemula. Server yang penulis gunakan berisi prosesor Core 2 Duo, RAM 1 GB, harddisk SATA 160 GB, dan ethernet card Gigabit onboard (Marvell) yang dikenali langsung oleh Ubuntu 7.04 saat instalasi.

Persiapan sebelum instal

Penulis menggunakan alamat IP yang tersedia di dalam jaringan. Untuk penerapan oleh pembaca, penomoran IP bisa disesuaikan. Hal-hal umum yang kita siapkan sebelum instalasi:

1. Domain e-mail. Dalam instalasi ini, penulis menggunakan sanjaya.com untuk domain e-mail. Jika Anda memiliki domain sendiri, dapat langsung digunakan. Zimbra melakukan pengecekan domain dengan hostname server. Jika server ini digunakan khusus internal, perlu dipersiapkan DNS record khusus IP internal.
2. Nama Workgroup/Domain. Dalam Windows server, *domain* merupakan workgroup yang memiliki otentikasi terpusat. Penulis menggunakan domain: BEKASI. Sebaiknya membuat Workgroup baru untuk server ini, agar proses berjalan dengan lancar.
3. Pembagian alamat IP. Server yang digunakan diberikan IP 172.20.1.17 (netmask 255.255.0.0). Nama server yang akan digunakan adalah PDC, dengan lengkapnya pdc.sanjaya.com.
4. Dalam tulisan ini, diasumsikan DNS dan DHCP server untuk LAN sudah terinstall dengan baik dan WINS server (untuk DHCP client) diarahkan ke IP 172.20.1.17. Jika belum terdapat DNS dan DHCP server lokal, semua *assign-*



Gambar 1. Mengedit file hosts.

ment IP diterapkan secara manual. Di dalam DNS server, A record pdc.sanjaya.org diarahkan ke IP 172.20.1.17, sedangkan MX record sanjaya.org diarahkan ke sanjaya.com.

5. List user yang akan menggunakan server ini. Ada tools yang bisa di-*download* untuk migrasi dari server Exchange ke Zimbra di Gallery Zimbra. Dalam tutorial ini, penulis tidak membahas *tools* tersebut.
6. Dalam melakukan instalasi penulis menggunakan otoritas root.

Instalasi Zimbra dan service pendukung

Dengan anggapan bahwa Ubuntu 7.04 (standar) sudah terinstal dan berjalan dengan baik, kita lanjutkan proses instalasi Zimbra dan Samba. Aplikasi Zimbra untuk Ubuntu kita download versi open source di *www.zimbra.com*, dan ambil versi 4.5.4 (jangan yang 4.5.5 karena ada ketidakcocokan dengan zimlet ZimbraSamba). Selain itu, download juga beberapa zimlet dari Gallery: Zimbra LDAP Utils Extensions, ZimbraPosixAccount, dan ZimbraSamba.

Ada permintaan khusus agar Zimbra bisa terinstal dengan baik:

1. Akan lebih baik kita berikan IP permanen untuk server ini.
2. Kita edit /etc/hosts, dan baris 127.0.1.1 kita remark. Zimbra akan menghentikan proses instalasi jika terdapat hosts dengan IP 127.0.1.1 ini. Perubahan ini kita kembalikan saat instalasi selesai. (Lihat gambar 1).
3. File /etc/lsb-release kita edit, ubah parameter di DISTRIB_RELEASE menjadi 6. Perubahan ini kita kembalikan saat instalasi selesai. (Lihat gambar 2). Penggantian ini diperlukan, karena instalasi Zimbra baru tersedia untuk versi 6.
4. Kita ekstrak dulu instalasi Zimbra for Ubuntu, dan dijalankan proses instalasinya. Kekurangan file instalasi dapat kita ketahui saat proses ini.

   Cara menjalankan dengan langkah-langkah berikut:

   - Ekstrak file instalasi Zimbra, zcs-4-.5.4 (dengan perintah tar atau menggunakan Archive Manager).
   - Masuk ke folder zcs.
   - Jalankan ./install.sh (Lihat gambar 3).

   Akan muncul file-file dibutuhkan yang belum ada (Lihat gambar 4).
5. Catat kekurangan file/library yang tampil, dan gunakan Synaptic atau apt-get di terminal untuk install file tersebut. Setelah library terinstal semua, ulangi langkah instalasi di poin 4 di atas. Berikut ini hal-hal yang harus diperhatikan:
   - Pada pertanyaan services yang harus diinstall, zimbra-ldap, zimbra logger, zimbra-mta, zimbra-snmp, zimbra-store, dan zimbra-spell, semuanya dijawab: Y (Yes).
   - The system will be modified. Continue [N] harus dijawab Y.
   - Setelah selesai instalasi, akan muncul pertanyaan:

```
DNS ERROR resolving MX for pdc.
sanjaya.com
. . .
Change domain name [Yes] : Y
Create domain : sanjaya.com
. . .
Done
```

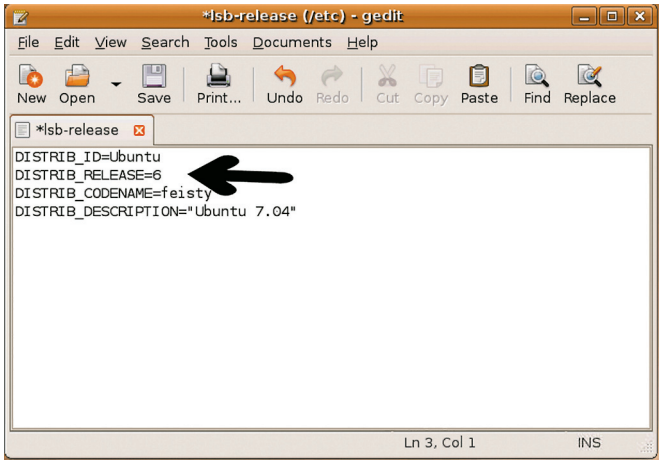
   - Menu instalasi akan muncul. Ada kewajiban untuk membuat password, yang bisa dilihat di menu dengan kode ******* (Lihat gambar 5 kiri atas). Untuk masuk, tekan 6 lalu tombol Enter dan tekan 4, lalu tombol Enter. Isilah password baru untuk administrasi Zimbra. Setelah selesai, tekan r dan [Enter] lalu a dan [Enter], Y dan [Enter]. Nama file config tidak perlu diubah.
   - Pertanyaan terakhir, apakah akan mengirimkan e-mail pemberitahuan ke admin Zimbra.
6. Kembalikan ke kondisi semula, dengan perubahan pada poin 2 dan 3.
7. Dalam terminal, dengan menggunakan user zimbra, kita ubah dulu password root LDAP server:

```
# su zimbra
$ zmldappasswd root [password baru]
$ zmcontrol stop
$ zmcontrol start
$ exit
```

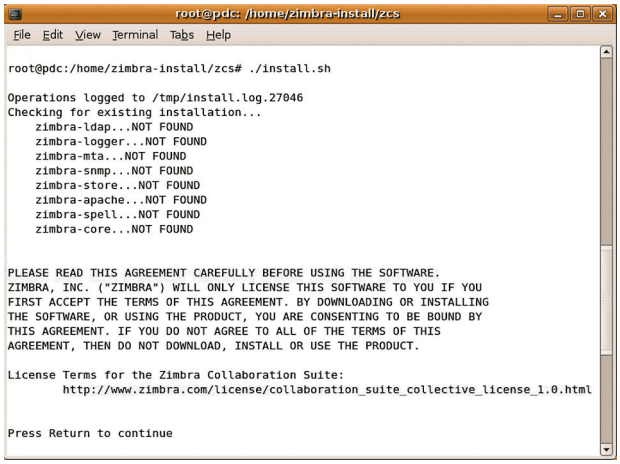
8. Install Zimbra LDAP Utils Extensions. Masuk ke folder /opt/zimbra/lib/ext dan buat folder zimbraldaputils. Ekstrak zimbraldaputils.jar dari zimbraldaputils.zip dan letakkan di folder yang baru dibuat tersebut.

   Restart server zimbra dengan urutan perintah di terminal:

```
· su zimbra
```



Gambar 2. Mengedit file lsb-release.



Gambar 3. Menjalankan script install.

```
· zmcontrol stop
· zmcontrol start
```

9. Ekstrak File ZimbraPosixAccount.zip ke desktop, dan edit file config_template.xml di dalam folder zimbra_posixaccount. Contoh ada di gambar 6.
   Sesuaikan dengan konfigurasi domain yang akan kita buat (sanjaya.com):

```
· ldapSuffix di isi dengan :
dc=sanjaya,dc=com
```

   Satukan dalam file zimbra_posixaccount.zip semua file di dalam folder zimbra_posixaccount termasuk config_template.xml yang kita edit tadi. File-file tersebut berada pada posisi paling atas di zimbra_posixaccount (bukan di bawah folder di bawahnya).
10. Lakukan hal yang sama dengan file ZimbraSamba.
11. Dari komputer tersebut, dengan menggunakan Firefox masuk ke web admin Zimbra, https://localhost:7071/zimbraAdmin (Lihat gambar 7).
    Masuk ke menu "Admin Extensions", lalu klik Deploy New. Upload zimbra_posixaccount.zip yang sudah diedit tadi, lalu klik Deploy. Lakukan hal yang sama dengan zimbra_samba.zip.
12. Edit file /etc/environment, tambahkan baris berikut:

```
LD_LIBRARY_PATH=/opt/zimbra/lib:/
usr/lib
```

    Karena kita belum berencana me-*restart* server ini, maka ketik di terminal:

```
# export LD_LIBRARY_PATH=/opt/
zimbra/lib:/usr/lib
```

    Untuk sementara, kita beralih untuk instal modul PAM-LDAP & NSS-LDAP, serta server Samba.

Instalasi PAM dan NSS LDAP

Dengan menggunakan terminal, ketik:

```
# apt-get install libpam-ldap
```

Akan muncul dialog, dan diminta untuk memasukkan beberapa parameter, antara lain:

- LDAP server, diisi dengan ldap://pdc.sanjaya.com/.
- Search base, diisi dengan domain e-mail : dc=sanjaya,dc=com.
- Version diisi dengan 3.
- Use local root database admin: Yes.
- Does the LDAP database require login: Yes.
- LDAP account for root: uid=zimbra, cn=admins,cn=zimbra.
- LDAP root access password, lihat password LDAP yang kita ganti saat instal zimbra di atas.
- Unprivileged database user : uid=zimbra, cn=admins,cn=zimbra.
- Password, sama seperti password root di atas.

Selanjutnya di terminal, ketik:

```
# apt-get install libnss-ldap
```

Pertanyaan yang muncul dijawab sama seperti instalasi libpam-ldap.

Konfigurasi Zimbra LDAP

Untuk mengintegrasikan Samba ke dalam LDAP, dibutuhkan samba.schema. File skema tersebut ada dalam paket instalasi samba-doc. Untuk mendapatkan file tersebut, kita install dulu Samba dan docnya dengan menggunakan perintah di terminal:

```
# apt-get install samba samba-doc
```

Setelah terinstal semua, masuk ke folder /usr/share/doc/samba-doc/examples/ LDAP, dan ekstrak file samba.schema di dalam samba.schema.gz ke folder /opt/zimbra/openldap/etc/openldap/shema.

Lalu edit file /opt/zimbra/conf/slapd.conf.in, tambahkan di bawah blok "include" yang ada:

```
include /opt/zimbra/openldap/etc/
openldap/schema/nis.schema
include /opt/zimbra/openldap/etc/
openldap/schema/samba.schema
```

Tambahkan juga di baris paling bawah:

```
# index untuk PAM
index uidNumber eq
index gidNumber eq
index memberUID eq

# index untuk Samba
index sambaSID eq
index sambaPrimaryGroupSID eq
index sambaDomainName eq
```

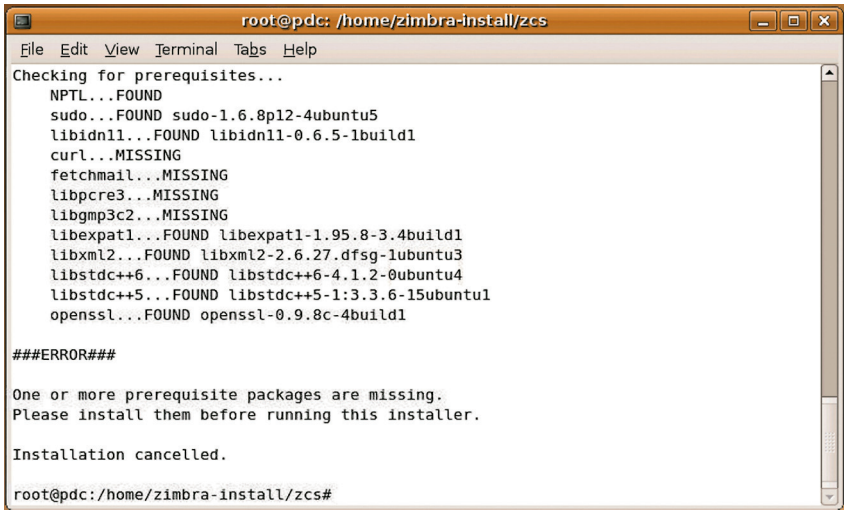
Setelah disimpan, jalankan perintah berikut sebagai user zimbra (su zimbra -):

```
$ zmprov mcf +zimbraAccountExtraObje
ctClass posixAccount
$ zmprov mcf +zimbraAccountExtraObje
ctClass sambaSamAccount
```

Lalu restart services zimbra dengan perintah zmcontrol stop / start seperti sebelumnya.

Instal dan konfigurasi Samba

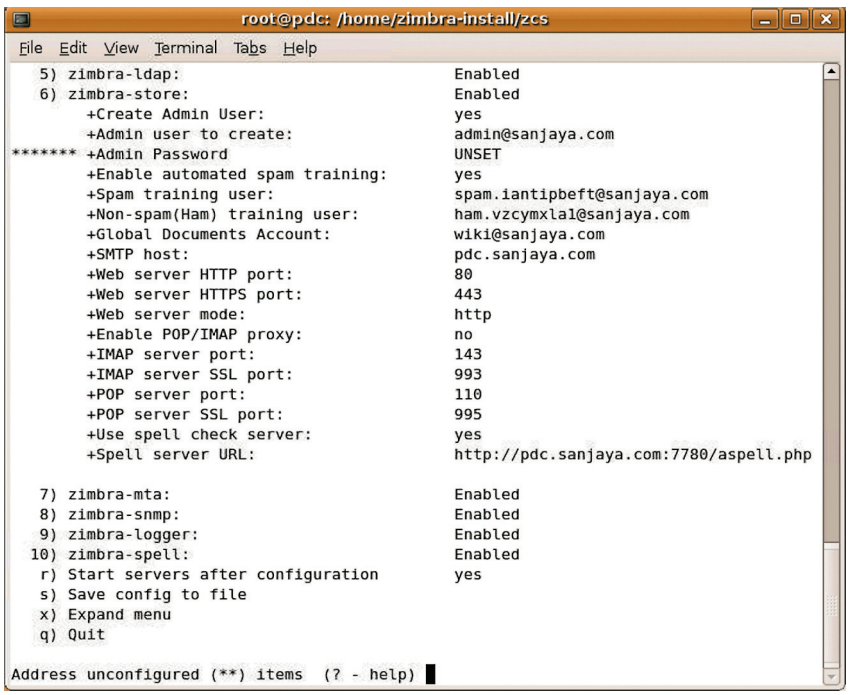
Ada banyak cara untuk mengonfigurasi Samba sesuai kebutuhan. Dalam pembuatan server ini, saya mengonfigurasikan Samba untuk menggunakan ZimbraLDAP untuk pengecekan password dan bertindak sebagai PDC untuk domain BEKASI dan sebagai WINS server untuk jaringan.



Gambar 4. Pesan kesalahan karena kekurangan file.

Berikut konfigurasi Samba dalam smb.conf:

```
[global]
workgroup = BEKASI
netbios name = PDC
winbind nested groups = no
os level = 33
preferred master = yes
enable privileges = yes
server string = %h server (Samba, Ubuntu Linux)
wins support =yes
dns proxy = no
name resolve order = wins bcast hosts
log file = /var/log/samba/log.%m
log level = 3
max log size = 1000
syslog only = no
syslog = 0
panic action = /usr/share/samba/panic-action %d
security = user
encrypt passwords = true
ldap passwd sync = yes
passdb backend = ldapsam:ldap://pdc.sanjaya.com/
ldap admin dn = uid=zimbra,cn=admins,cn=zimbra
ldap suffix = dc=sanjaya,dc=com
ldap group suffix = ou=groups
ldap user suffix = ou=people
ldap machine suffix = ou=machines
obey pam restrictions = no
passwd program = /usr/bin/passwd %u
passwd chat = *Enter\snew\sUNIX\spassword:* %n\n *Retype\snew\sUNIX\spassword:* %n\n *password\supdated\ssuccessfully* .
domain logons = yes
logon path = \\%N\departemen\profile
logon home = \\%N\departemen
logon script = logon.bat
add user script = /usr/sbin/adduser --quiet --disabled-password --gecos %u
add machine script = /usr/sbin/adduser --shell /bin/false --disabled-password --quiet --gecos machine account --force-badname %u
socket options = TCP_NODELAY
domain master = yes
local master = yes

[homes]
comment = Home Directories
browseable =no
read only = No
valid users = %S

[departemen]
comment = Folder Departemen
path = /home/share/group/%G
share modes = yes
locking = no
read only = No

[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon
guest ok = yes
locking = no
share modes = no

[profiles]
comment = Users profiles
path = /var/lib/samba/profiles
read only = No

[profdata]
comment = Profile Data Share
path = /var/lib/samba/profdata
read only = No
```



Gambar 5. Tanda belum membuat password.

```
profile acls = Yes

[share]
comment = Tempat Tukar-Menukar Files
writable = yes
create mask = 0664
directory mask = 0775
path = /home/share/everyone

[printers]
comment = All Printers
browseable = no
path = /tmp
printable = yes
public = no
writable = no
create mode = 0700

[print$]
comment = Printer Drivers
path = /var/lib/samba/printers
browseable = yes
read only = yes
guest ok = no
```

Simpan konfigurasi di atas. Agar Samba bisa berkomunikasi dengan LDAP, kita harus menyimpan password root LDAP di Samba dengan perintah:

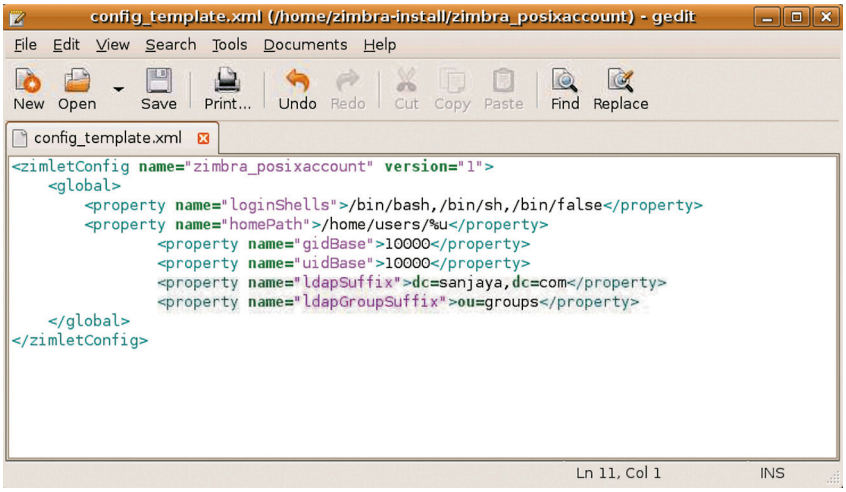
```
# smbpasswd w passwordbaru
```

Passwordbaru di atas, merupakan password root dari ZimbraLDAP yang telah kita simpan pada proses instalasi di atas.

Restart kembali services zimbra dengan perintah zmcontrol stop / start seperti sebelumnya, lalu restart service Samba:

```
/etc/init.d/samba restart
```

Login ke web admin Zimbra, masuk ke menu Samba Domain. Jika instalasi sukses, domain (BEKASI) yang kita buat akan muncul di list. Lihat gambar 8.



Gambar 6. Mengedit file config_template.

Konfigurasi pam_ldap dan nss_ldap

Kita perlu mengubah konfigurasi pam dan nss, agar server linux secara otomatis melakukan pengecekan ke server LDAP (Zimbra) saat ada akses ke userID dan password.

Edit file /etc/libnss-ldap.conf, pastikan konfigurasinya sudah benar (dengan asumsi password LDAP adalah "passworbaru").

```
base dc=sanjaya,dc=com
host pdc.sanjaya.com
binddn uid=zimbra, cn=admins,
cn=zimbra
bindpw passwordbaru
rootbinddn uid=zimbra, cn=admins,
cn=zimbra
```

Samakan konfigurasi di file /etc/pamd_ldap.conf dengan /etc/libnss-ldap.conf di atas.

Lihat /etc/libnss-ldap.secret, dan pastikan bahwa password tertulis dengan benar. Copy file tersebut ke /etc/pam_ldap.secret.

Edit file /etc/nsswitch.conf, ganti baris berikut:

```
passwd: compat
group: compat
dengan
passwd files ldap
group files ldap
```

Perubahan tersebut akan membuat nsswitch untuk menggunakan LDAP saat mencari uid dan gids.

Ubah /etc/pam.d/common-account, sehingga terbaca sebagai berikut:

```
account sufficient pam_unix.so
account sufficient pam_ldap.so
```

Ubah /etc/pam.d/common-auth, sehingga terbaca sebagai berikut:

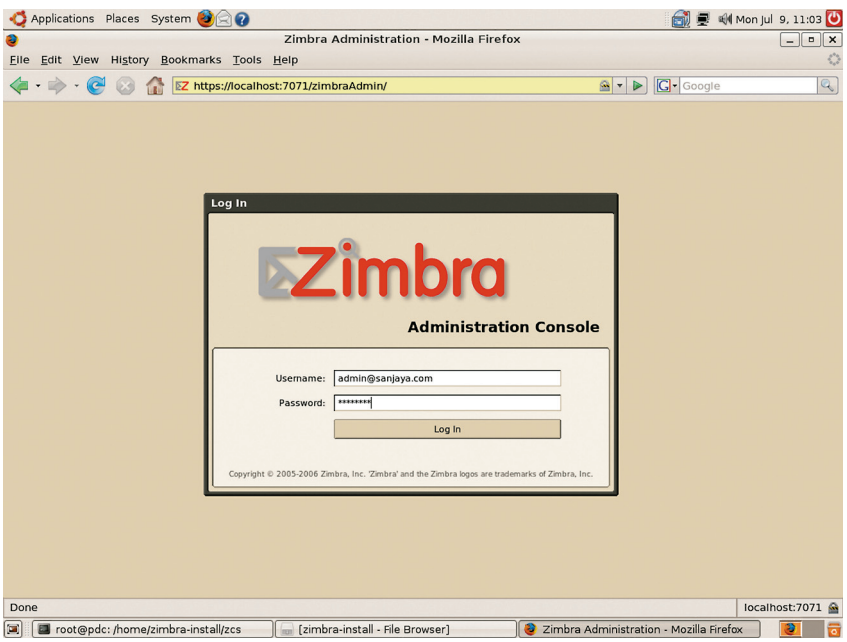
```
auth sufficient pam_unix.so
auth sufficient pam_ldap.so
```

Ubah /etc/pam.d/common-password, sehingga terbaca sebagai berikut:

```
password sufficient pam_unix.so
password sufficient pam_ldap.so
```

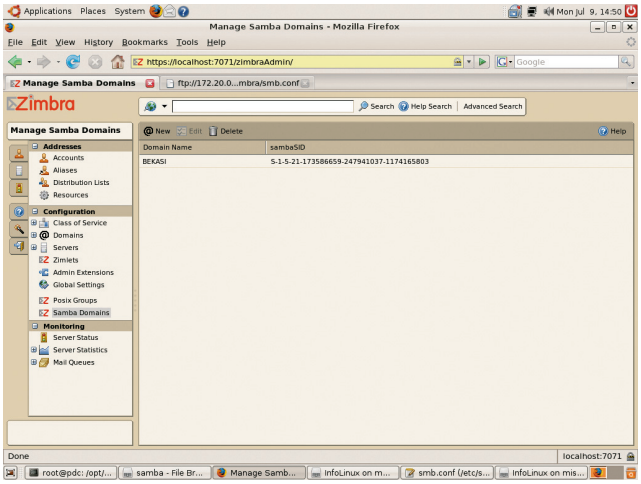
Ubah /etc/pam.d/common-session, sehingga terbaca sebagai berikut:

```
session sufficient pam_unix.so
session sufficient pam_ldap.so
```

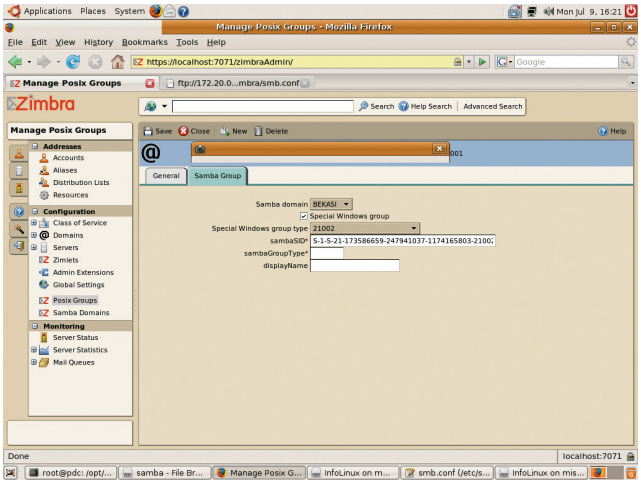


Gambar 7. Halaman login ke Zimbra Admin.

IKLAN



Gambar 8. Contoh domain BEKASI.



Gambar 9. Samba Group type.

Administrasi domain

ZimbraSamba dapat kita tes dengan membuat Group (Posix) lebih dulu. (Lihat gambar 9). Untuk SambaGroupType, isi dengan 2 sebagai angka default.

Untuk mengecek apakah group dalam domain Samba sudah terintegrasi dengan baik, ketik di terminal dengan privilege root:

```
# getent group
```

Group yang tadi dibuat, seharusnya ikut terlihat dengan perintah tersebut.

Selanjutnya, kita buat user, ke menu Account, klik New. Isi semua *field* yang dibutuhkan, dan lanjutkan dengan Next. Setelah layar Advance, akan muncul dua layar baru, yaitu Posix dan Samba Domain. Setelah kita klik Finish, user baru dapat dicek di terminal dengan mengetik perintah:

```
# getent passwd
```

Untuk folder Home, kita buat secara manual di server, sesuai dengan profile yang telah ditentukan di halaman Samba saat pembuatan user. Selanjutnya, masuk ke web zimbraAdmin, hapus alias root@sanjaya.com, lalu ketik di terminal:

```
smbpasswd -a root
```

Langkah berikutnya, membuat group "Domain Admins" menggunakan web zimbraAdmin, pada pembuatan group posix di bagian Samba pilih "Special Windows" group type "Domain Admins". Lalu sebagai root di terminal, ketik perintah:

```
net rpc rights grant "BEKASI\
Domain Admins" SeAddUserPrivilege
SeMachineAccountPrivilege
SePrintOperatorPrivilege
```

Menambahkan komputer berbasis Windows ke domain

Pada komputer Windows (NT/2000/XP), login sebagai administrator lokal, dan gabung ke domain Samba ini seperti penggabungan biasa. Gunakan user yang anggota "Domain Admins" untuk verifikasi saat pendaftaran ini.

Setelah proses pendaftaran selesai, bisa dilihat dengan perintah ldapsearch. Contoh jika nama komputer clientnya BEKCLIENT001:

```
 /opt/zimbra/openldap/bin/
ldapsearch -h bekasi | grep
bekclient
```

Seharusnya muncul jawaban kurang lebih sebagai berikut:

```
 # bekclient001$, machines,
sanjaya.com
 dn: uid=bekclient001$,ou=machines,
dc=sanjaya,dc=com
 uid: bekclient001$
```

Dengan munculnya nama komputer tersebut server sudah siap digunakan. Jika ingin menambahkan fasilitas Zimbra, dapat download Zimlet di Gallery Zimbra. Koneksi email bisa menggunakan POP3/IMAP atau webmail *http://pdc.sanjaya.com* yang lebih lengkap fasilitasnya.

Jika kita ingin membuat komputer lain yang berbasis Linux untuk menggunakan *account* yang sama, setara dengan client Windows yang lain, ada beberapa *setting* yang harus kita terapkan. Instal PAM dan NSS LDAP pada client atau server ini. Setelah itu, kita samakan juga konfigurasi pam_ldap dan nss_ldap seperti pembahasan sebelumnya. Saat kita restart komputer ini, account yang terdapat di Zimbra sudah bisa digunakan di komputer baru ini.

Back-up dan restore Zimbra

Zimbra yang kita install ini merupakan versi OpenSource. Untuk versi ini Backup dan Restore tidak ada dalam menu administrasi Zimbra. Kita dapat melakukan back-up secara manual dengan cara sebagai berikut:

```
# su zimbra -
$ zmcontrol stop
$ exit
```

Perintah di atas untuk menghentikan dulu server Zimbra.

```
# cp -rp /opt/zimbra [lokasi backup]
```

Ini merupakan perintah copy folder zimbra. Lokasi back-up yang digunakan sesuai dengan otoritas user yang dipakai. Besarnya *space* back-up disesuaikan dengan besarnya space yang digunakan zimbra. Setelah copy selesai, hidupkan kembali services Zimbra:

```
# su zimbra -
$ zmcontrol start
$ exit
```

Hasil *copy*-an di [lokasi backup] tersebut bisa kita mampatkan dulu dengan perintah tar, atau langsung ditulis ke tape backup jika Anda memilikinya.

Untuk restore-nya, pastikan bahwa versi yang sama telah terinstall dengan baik. Setelah siap untuk restore, matikan dulu server Zimbra dengan perintah zmcontrol stop. Gunakan perintah cp untuk mengkopi backup zimbra ke folder /opt/zimbra.

Jika restore sudah selesai tapi ada masalah dengan permission, sebagai user zimbra gunakan perintah:

```
$ /opt/zimbra/libexec/zmfixperms
```

Hal-hal lain yang berkaitan dengan Zimbra, bisa dilihat di *http://wiki.zimbra.com*

Indra Sanjaya [indsan@gmail.com]

Penjadwalan Tugas Menggunakan Cron dan AT

Pada suatu waktu, mungkin Anda membutuhkan sistem bekerja pada tengah malam ketika penggunaan sistem dalam kondisi tidak sibuk. Atau Anda menginginkan suatu pekerjaan yang dilakukan secara harian atau mingguan secara otomatis? Jika ya, gunakan saja aplikasi cron dan at yang memiliki kemampuan untuk penjadwalan tugas secara periodik.

Tugas menjadi seorang sistem administrator, terkadang banyak berhubungan dengan penjadwalan tugas yang harus dijalankan secara berkala dalam kurun waktu tertentu pada sistem Linux. Salah satu contoh pekerjaan ini, di antaranya adalah pekerjaan merotasi file log sehingga kapasitas filesystem menjadi tidak penuh, *back-up* data, dan konek ke time server untuk menjaga agar sistem time Anda selalu di sinkronisasi.

Tugas lain yang tidak ada kaitannya dengan sistem, terkadang juga membutuhkan utiliti penjadwalan tugas yang dapat menjalankan tugas pada kurun waktu tertentu. Misal, jika ingin men-*download* suatu file yang lumayan besar pada saat load sistem Internet sedang tidak ada yang menggunakan.

Untuk melakukan hal ini, sudah tersedia sejumlah utiliti di Linux yang berfungsi untuk melakukan hal ini. Salah satu yang paling banyak digunakan dan cukup andal untuk kebutuhan ini adalah cron dan at. Kedua aplikasi ini biasa dikemas secara *default* oleh kebanyakan distro Linux, karena kemudahan penggunaan dan kestabilan aplikasi ini. Utilitas Cron yang dibuat oleh Paul Vixie ini, dapat menjalankan suatu perintah sesuai dengan waktu dan hari yang telah ditentukan. Dengan ini, Anda tidak perlu lagi menunggu untuk sekadar menjalankan suatu perintah pada waktu yang Anda inginkan.

Utiliti at yang dibuat oleh Thomas Koenig, juga dapat digunakan untuk menunggu, menjalankan, atau menghapus suatu tugas, untuk kemudian dieksekusi. Perbedaannya dengan cron, at lebih cocok digunakan untuk kebutuhan penjadwalan yang dilakukan dalam tempo satu kali saja. Hal ini berbeda dengan cron yang ditujukan untuk penjadwalan tugas yang dilakukan secara berulang.

Pada tutorial kali ini, akan dibahas beberapa hal yang berkaitan dengan penjadwalan tugas di Linux dengan menggunakan crontab dan at.

Menjalankan tugas setiap kurun waktu tertentu

Untuk dapat menjalankan tugas dalam interval waktu tertentu, kita dapat menggunakan utiliti cron, yang terdiri dari crond daemon dan sekumpulan tabel yang mendeskripsikan apa yang akan dilakukan dan berapa lama frekuensi itu berlangsung. Daemon akan berjalan setiap waktu dan menngecek crontabs untuk menentukan apa yang dibutuhkan. Para pengguna dapat memanajemen crontab dengan menggunakan perintah crontab. Daemon crond biasanya dapat berjalan dengan menggunakan proses init pada saat startup sistem.

Sebagai latihan, dalam contoh berikut kita akan membuat sebuah bash script sederhana, yang kemudian kita gunakan crontab untuk mensetup cron job. Skrip ini dapat menampilkan hari dan waktu yang sedang berjalan pada saat itu, sehingga pada saat cron berjalan, kita akan mengetahui *output* dari skrip ini. Mengonfigurasi entiti crontab membutuhkan string dengan escaped shell metacharacter, sehingga cara ini lebih mudah dilakukan dengan menggunakan parameter dan perintah yang simpel. Dalam contoh ini, perintah echo akan menjalankan skrip yang terdapat pada /home/supriyanto/mycrontab.sh, tanpa menggunakan parameter. Berikut listing mycrontab.sh.

```
#!/bin/bash
 echo "It is now $(date +%T) on
$(date +%A)"
```

Membuat sebuah crontab

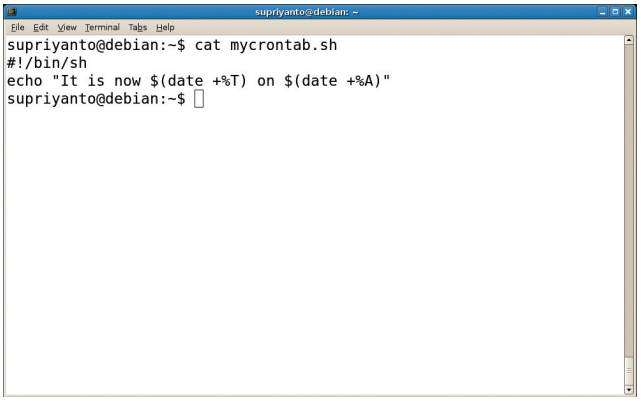
Untuk membuat sebuah crontab, Anda dapat menggunakan perintah crontab ditambah dengan option -e (untuk "edit") option. Perintah tersebut secara otomatis akan membuka editor vim, jika Anda tidak mendefinisikan editor lainnya di *environment* variabel EDITOR atau VISUAL.

Setiap entri yang terdapat pada crontab terdiri atas enam bagian, yaitu:

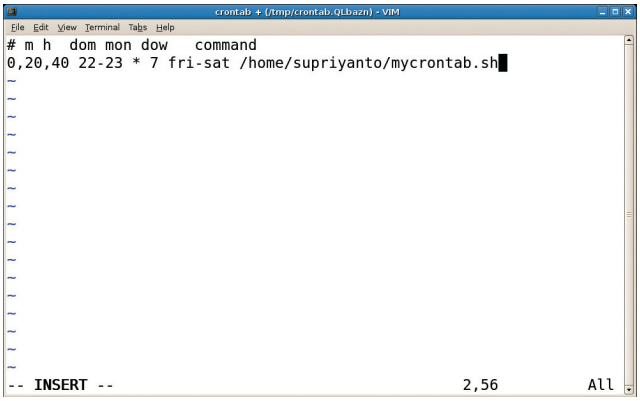
1. Menit.
2. Jam.
3. Hari dalam sebulan.
4. Bulan dalam setahun.
5. Hari dalam seminggu.
6. String atau perintah yang akan dieksekusi oleh sh/bash.

Penjelasan masing-masing field:

- Menit dan jam berkisar antara 0-59 dan 0-12.
- Entri hari dalam sebulan dan bulan dalam setahun, berkisar antara 1-31 dan 1-12.
- Hari dalam seminggu berada dalam kisaran 0-6, di mana 0 adalah Minggu.



Isi skrip mycrontab.sh.



Mengedit file crontab untuk menjalankan skrip mycrontab.sh.

- Hari dalam seminggu, dapat juga di tulis sebagai sun, mon, tue, dan seterusnya.
- Field keenam adalah inti dari entri ini. Field ini mendefinisikan string yang selanjutnya akan dijalankan oleh sh. Tanda persen (%) di terjemahkan sebagai ganti baris. Jadi jika Anda tetap menginginkan penggunaan a% atau karakter spesial lainnya, Anda dapat memisahkannya dengan menggunakan backslash (\). Baris yang mengandung % pertama akan di teruskan ke shell, di mana semua baris setelah % akan diteruskan sebagai standar input.

Period

Beragam waktu yang berhubungan dengan fields, juga dapat menspesifikasikan individual value, range of value, seperti 0-10 atau sun-wed, atau daftar pemisahan koma dari jangkauan dan nilai individual. Cara mendefinisikan entry crontab untuk perintah yang telah kita buat sebelumnya, akan terlihat seperti baris berikut:

```
0,20,40 22-23 * 7 fri-sat /home/
supriyanto/mycrontab.sh
```

Dalam contoh di atas, perintah akan dijalankan setiap menit ke 0, 20, dan 40 (setiap 20 menit), antara jam 10-11 malam di setiap hari Jumat dan Sabtu yang terdapat selama bulan Juli. Anda dapat melihat option lengkap crontab di manual page crontab.

Bagaimana dengan output-nya?

Setelah mengikuti perintah di atas, dan memasukkan perintah tersebut ke dalam crontab, mungkin sebagian Anda ada yang bertanya dimana hasil outputnya? Kebanyakan perintah di desain agar dapat menggunakan cron dengan kemampuan menghasilkan output log menggunakan syslog. Setiap output akan di berikan ke stdout untuk di kirimkan ke mail user yang bersangkutan. Berikut contoh output cron yang dihasilkan.

```
From root@debian.example.com  Wed
Aug  1 03:40:01 2007
X-Original-To: supriyanto
Subject: Output from your job
1
To: supriyanto@debian.example.com
Date: Wed,  1 Aug 2007 03:40:00
+0700 (WIT)
From: root@debian.example.com (root)

It is now 03:40:00 on Wednesday
```

Crontab yang Anda buat dengan perintah crontab, akan disimpan pada direktori /etc/spool/cron, dalam folder nama user yang membuatnya. Jadi pada contoh diatas, hasilnakan disimpan pada disimpan pada direktori /etc/spool/cron/supriyanto.

/etc/crontab

Sebagai tambahan file user crontab yang terdapat di /var/spool/cron, cron juga mengecek /etc/crontab dan file-file yang terdapat dalam direktori /etc/cron.d. Sistem crontab ini, memiliki satu tambahan field antara entry field kelima dan keenam. Tambahan field ini menspesifikasikan nama user yang menjalankan perintah tersebut, yang biasanya adalah user root. Isi file /etc/crontab mungkin akan terlihat seperti contoh berikut:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.
hourly
02 4 * * * root run-parts /etc/cron.
daily
22 4 * * 0 root run-parts /etc/cron.
weekly
42 4 1 * * root run-parts /etc/cron.
monthly
```

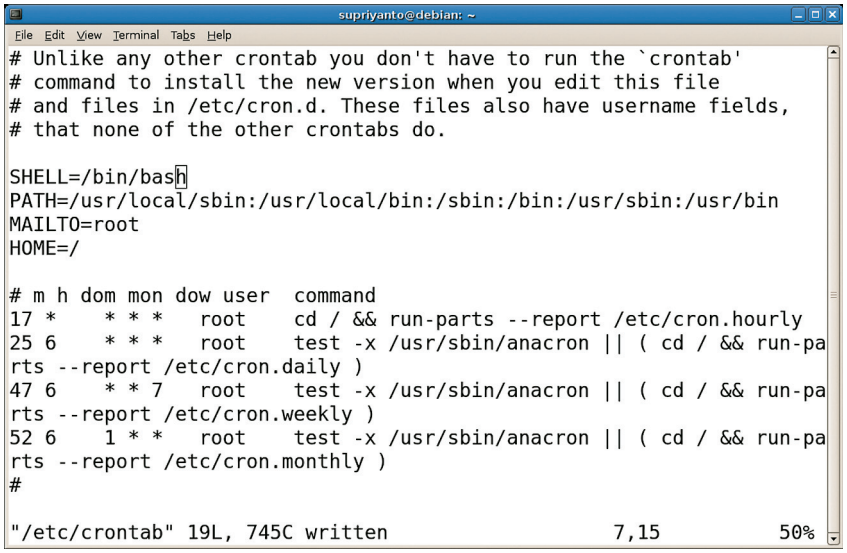
Pada contoh ini, terlihat jelas kalau pekerjaan sebenarnya dilakukan oleh perintah run-parts, dimana skrip tersebut menjalankan /etc/cron.hourly, /etc/cron.daily, dan sebagainya. File /etc/crontab dapat dengan mudah mengontrol waktu untuk menjalankan tugas. Crontab juga dapat menyertakan *shell variables assigment* yang dapat di set sebelum perintah dijalankan.

Anacron

Cron hanya dapat bekerja dengan baik pada sistem yang berjalan secara berkelanjutan. Untuk sistem yang sering kali dimatikan, seperti misal laptop, aplikasi lain yang bernama anacron dapat menangani penjadwalan tugas yang dilakukan perhari, perminggu, atau perbulan dengan menggunakan utilitas cron. Anacron tidak dapat menangani tugas per-jam.

Anacron meletakan file timestamp pada /var/spool/anacron untuk mencatat setiap tugas yang sedang berjalan. Saat anacron berjalan, aplikasi ini akan memeriksa apakah masih terdapat sejumlah tugas pada hari itu, dan menjalankan kembali tugas itu jika memang diperlukan. Tabel dari pekerjaan yang dilakukan oleh anacron disimpan dalam file /etc/anacrontab, yang memiliki perbedaan format dengan file /etc/crontab. Sama halnya dengan file /etc/crontab, file /etc/anacrontab, juga menyediakan beberapa parameter konfigurasi. Setiap tugas terdiri atas empat field, yakni:

1. Period.
2. Delay.
3. Job Identifier.
4. Command.



Isi file /etc/crontab.

Penjelasan masing-masing field:

- Period merupakan angka dalam hari, tetapi mungkin hanya dispesifikasikan sebagai @monthly untuk memastikan bahwa suatu tugas berjalan hanya sekali dalam sebulan.
- Delay merupakan angka dalam hitungan menit, untuk menunggu setelah tugas dilakukan dengan tujuan untuk menjalankan tugas sebenarnya sebelum memulainya. Hal ini cukup berguna untuk menangani banyaknya rugas dalam satu waktu, ketika tugas sebelumnya sedang berjalan.
- Job identifier dapat menyediakan beberapa blank character selain karakter slash (/).
- Command merupakan perintah yang akan di jalankan dalam waktu yang telah ditentukan oleh perintah at.

Baik file /etc/crontab maupun /etc/anacrontab akan selalu di update setiap kali di edit langsung. Kita juga tidak dapat menggunakan perintah crontab untuk mengupdate file ini, atau file yang terdapat pada direktori /etc/cron.d.

Menjalankan tugas pada waktu yang ditentukan

Terkadang kita menginginkan untuk menjalankan suatu tugas hanya berlangsung sekali, daripada berulang-ulang. Untuk tujuan ini, kita dapat menggunakan perintah at. Perintah ini akan berjalan setelah membaca file yang telah dispesifikasikan dengan option -f, atau dari stdin, jika -f tidak digunakan. Sedangkan, option -m digunakan untuk mengirimkan email ke user ketika tidak terdapat stdout dari perintah yang dijalankan. Option -v menampilkan waktu ketika tugas akan dijalankan sebelum membaca tugas. Waktu juga dapat ditampilkan sebagai output.

Perintah berikut akan menjalankan skrip mycrontab.sh yang Anda buat sebelumnya.

```
$ at -f mycrontab.sh -v 04:31
Wed Aug  1 04:31:00 2007
```

Output tugas yang dilakukan oleh at:

```
Message 1:
From supriyanto@debian.example.com
Wed Aug  1 04:31:00 2007
X-Original-To: supriyanto
Subject: Output from your job
5
To: supriyanto@debian.example.com
Date: Wed,  1 Aug 2007 04:31:00
+0700 (WIT)
From: supriyanto@debian.example.com
(supriyanto)

It is now 04:31:00 on Wednesday
```

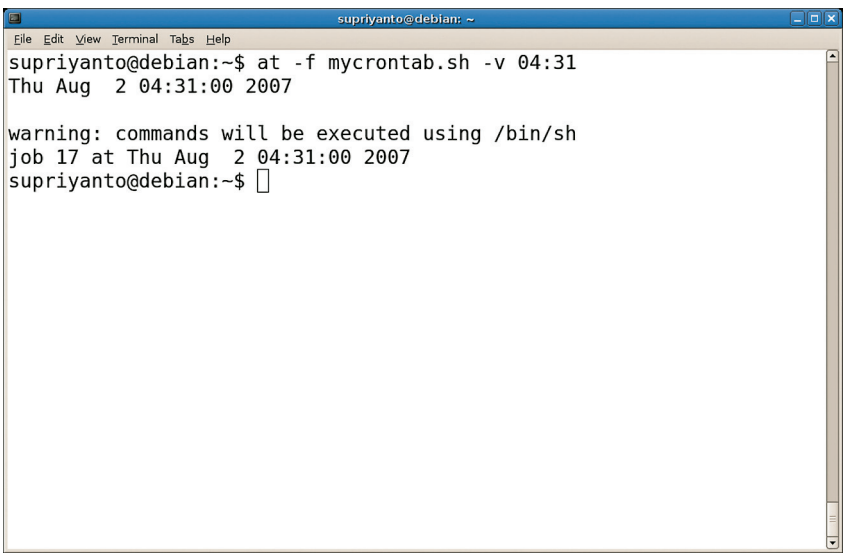
Selain menggunakan spesifikasi waktu seperti contoh di atas, kita juga dapat menggunakan spesifikasi waktu yang lebih kompleks seperti contoh berikut.

```
$ at -f mycrontab.sh 10pm tomorrow
job 7 at Thu Aug  2 22:00:00 2007

$ at -f mycrontab.sh 2:00 tuesday
job 8 at Tue Aug  7 02:00:00 2007

$ at -f mycrontab.sh 2:00 july 11
job 9 at Fri Jul 11 02:00:00 2008

$ at -f mycrontab.sh 2:00 next week
job 10 at Wed Aug  8 02:00:00 2007
```

Perintah at juga memiliki option -q, untuk meningkatkan waktu queue sehingga menambah hasil yang baik bagi suatu tugas. Selain perintah at, terdapat juga sejumlah perintah batch yang lain, yang hanya dapat menjalankan tugas pada saat load sistem sedang rendah.

Manajemen penjadwalan tugas

Pada bagian ini kita akan membahas sejumlah manajemen penjadwalan tugas berbasis cron maupun at, seperti cara menampilkan daftar tugas, cara menghapus suatu tugas yang



Membuat penjadwalan tugas skrip mycrontab.sh menggunakan at.

masih terdapat dalam daftar tugas, dan konfigurasi akses user untuk penjadwalan tugas.

Menampilan daftar tugas

Kita dapat memanajemen tugas yang terdapat pada cron dan at. Gunakan perintah crontab dengan option -l untuk melihat daftar tugas yang terdapat pada crontab. Anda juga dapat menggunakan perintah atq untuk menampilkan daftar tugas yang sedang queue saat menggunakan perintah at. Berikut contoh tampilan daftar tugas yang masih terdapat pada crontab.

```
$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/
supriyanto/mycrontab.sh

$ atq
6       Thu Aug  2 22:00:00 2007 a
supriyanto
9       Fri Jul 11 02:00:00 2008 a
supriyanto
10      Wed Aug  8 02:00:00 2007 a
supriyanto
8       Tue Aug  7 02:00:00 2007 a
supriyanto
7       Thu Aug  2 22:00:00 2007 a
supriyanto
```

Jika Anda ingin melihat perintah sebenarnya yang telah terdapat dalam jadwal yang akan dieksekusi oleh at, Anda dapat menggunakan perintah at dengan option -c dan job number. Anda akan diberi peringatan lingkungan kerja yang aktif pada saat menjalankan perintah at, dan kejadian yang terdapat akan disimpan dalam daftar tugas.

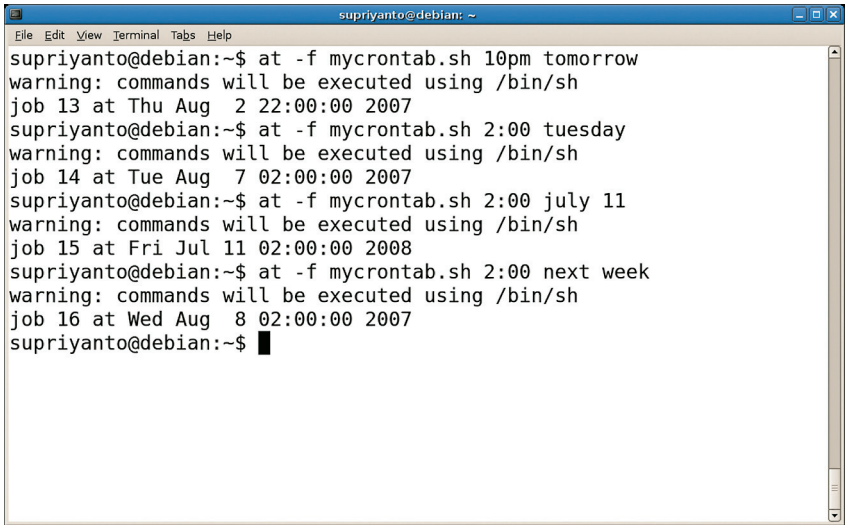
Menghapus daftar tugas

Kita dapat menghapus semua jadwal tugas cron dengan menggunakan option -r yang terdapat pada perintah cron.

Berikut langkah menghapus suatu tugas yang masih terdapat di cron.

```
$ crontab -l
# m h  dom mon dow   command
0,20,40 22-23 * 7 fri-sat /home/
supriyanto/mycrontab.sh

$ crontab -r

$ crontab -l
no crontab for supriyanto
```

Untuk menghapus sistem cron atau tugas yang terdapat pada anacron, Anda dapat mengedit file /etc/crontab, /etc/anacrontab, atau mengedit/menghapus file yang terdapat pada direktori /etc/cron.d.



Beberapa pendefinisian waktu di at yang lebih kompleks.

Anda juga dapat menghapus satu atau lebih tugas yang telah dijalankan sebelumnya dengan menggunakan perintah at, dengan menggunakan perintah atrm dan angka tugas yang sedang dijalankan. Penghapusan beberapa tugas dapat dipisahkan dengan menggunakan spasi. Berikut contoh penghapusan tugas dengan menggunakan atrm.

```
$ atq
6       Thu Aug  2 22:00:00 2007 a
supriyanto
9       Fri Jul 11 02:00:00 2008 a
supriyanto
10      Wed Aug  8 02:00:00 2007 a
supriyanto
8       Tue Aug  7 02:00:00 2007 a
supriyanto
7       Thu Aug  2 22:00:00 2007 a
supriyanto

$ atrm 8 9 10

$ atq
6       Thu Aug  2 22:00:00 2007 a
supriyanto
7       Thu Aug  2 22:00:00 2007 a
supriyanto
```

Hasil output diatas menunjukkan, kalau tugas nomor 8, 9, dan 10 yang terdapat pada at sudah dihapus dari daftar tugas, sehingga hanya tersisa tugas nomor 6 dan 7.

Konfigurasi akses user untuk penjadwalan tugas

Jika terdapat file /etc/cron.allow, setiap user selain root harus terdaftar agar dapat menggunakan utiliti cron dan crontab. Jika /etc/cron.allow tidak ada, tapi /etc/cron.deny ada, maka user selain root yang terdaftar di sana tidak akan dapat menggunakan aplikasi cron ataupun crontab. Jika kedua file itu ada, hanya super user yang diizinkan untuk dapat menggunakan cron dan crontab. Jika file /etc/cron.deny kosong, maka semua user diijinkan untuk dapat menggunakan utiliti cron secara default. Keterkaitan antara file /etc/at.allow dan /etc/at.deny memiliki pengaruh yang sama seperti cron untuk penggunaan utiliti at.

Demikian tutorial singkat mengenai penjadwalan tugas menggunakan cron dan at. Kedua aplikasi open source ini merupakan aplikasi yang hebat, dan cukup handal untuk digunakan. Tidak heran jika hampir semua platform berbasis Unix, memaketkan kedua aplikasi ini pada paket distribusinya.

Hampir semua distro Linux, juga menggunakan secara default aplikasi ini untuk penjadwalan tugasnya. Kebanyakan aplikasi di Linux yang membutuhkan penjadwalan tugas untuk dapat berjalan pada waktu tertentu, juga menggunakan aplikasi crontab untuk kebutuhan aplikasinya. Untuk lebih mempermudah penggunaan aplikasi ini, dokumentasi berbantuk manual pages yang sudah disertakan pada kemasan paket ini, sudah sangat mencukupi jika Anda ingin mengenal lebih lanjut sejumlah parameter atau option lain yang terdapat pada aplikasi cron dan at. Tinggal diperlukan semangat untuk mempelajari lebih dalam penggunaan kedua aplikasi ini. Akhir kata, selamat mencoba!

Supriyanto [supriyanto@infolinux.co.id]