Membuat Utility mass renamer Sendiri

Punya banyak file gambar dengan aneka nama yang tidak beraturan? Ingin merapikannya dengan mudah dan cepat? Gunakan saja berbagai *utility* untuk mengubah nama file. Tidak punya? Buat saja sendiri. Kita akan membahasnya di tulisan ini.

Kita seringkali menyimpan berbagai file di satu direktori dengan tidak rapi, dari sisi nama file. Terutama file-file hasil foto, atau berbagai koleksi gambar lainnya. Ketika jumlah file masih sedikit, kita mungkin masih bisa mengubah nama file gambar agar tampak lebih rapi. Tapi, kalau jumlah file sudah mencapai ribuan, biasanya rasa malas akan muncul.

Di tulisan ini, kita akan membuat sebuah utility *mass renamer* sederhana, yang dapat mengubah nama file dengan cepat dan terpola. Dengan demikian, aneka nama file kita yang mencapai ribuan bisa diubah menjadi gambar0001, gambar0002, dan seterusnya. Tentu saja, *prefix* nama file dan panjang *padding* nomor urut bisa disesuaikan.

Program akan kita bangun menggunakan *shell script*, memanfaatkan *tool* standar sistem. Dengan demikian, program dapat berjalan pada berbagai distribusi Linux, bahkan di Windows, misalnya dengan bantuan *cygwin*.

Tulisan ini dibangun di atas sistem slackware 11, namun seharusnya dapat diterapkan pada sistem lain tanpa permasalahan. Shell yang digunakan adalah *bash*.

Sebagai contoh, kita memiliki sebuah direktori dengan nama TEST, yang mengandung beberapa file contoh berikut:

```
$ ls -1 TEST/
gambar\ 1.png
mobil\ keren\ (dapat\ dari\ teman).PNG
new\ image\ 123.JPG
pemandangan\ pantai\ 1.jPG
```

Nantinya, semua gambar di direktori TEST tersebut akan:

- dikopikan ke direktori TEST.OUT
- diberi nama dengan pola nama file IMAGEXXXXX, di mana XXXXX menunjukkan nomor file, dilengkapi dengan padding angka 0. Contoh: IMAGE00001, IMAGE00002, dan seterusnya.
- Ekstensi nama file akan dipertahankan, dan akan ditambahkan ke nama file yang baru.

Setelah proses berlangsung dengan baik, direktori TEST bisa dihapus, dan direktori TEST.OUT bisa diganti nama menjadi TEST.

Berikut ini kita akan melihat *source code* nmass-renamer, utility yang akan kita bangun kali ini:

```
#!/bin/sh

#nmass-rename
#simple mass rename program
#http://www.noprianto.com/code.php


if [ $# -ne 4 ]
then
        echo "nmass-rename (c) Noprianto, 2007. GPL."
        echo "http://www.noprianto.com/code.php"
        echo
        echo "$0 <input_directory> <output_directory> <prefix> <length>"
        echo "example: $0 TESTDIR TESTDIR.OUT IMAGE 5 "
        echo " result: TESTDIR.OUT/IMAGE00001, TESTDIR.OUT/IMAGE00002, ...."
        echo
        exit 1
fi

if [ ! -d "$2" ]
then
        mkdir -p "$2"
fi

COUNTER=1
for f in $1/*
do
        NEW_FN=`printf "%s/%s%0$4d" $2 $3 $COUNTER`
        NEW_EXT=`echo ${f##*.}`
        NEW_FILE="$NEW_FN.$NEW_EXT"
        echo "Copying $f to $NEW_FILE"
        cp -a "$f" "$NEW_FILE"
        COUNTER=$(($COUNTER+1))
done

echo "done"
```

Berikanlah hak akses *executable*, dengan perintah berikut:

```
$ chmod +x nmass-rename
```

Program ini akan menerima empat

argumen:

- 1: direktori tempat, berisikan file-file yang ingin diubah namanya.
- 2: direktori *output*. Apabila tidak ditemukan, akan dibuat terlebih dahulu secara otomatis.
- 3: prefix nama file baru. Sebagai contoh: IMAGE, atau GAMBAR, atau FILE dan lainnya
- 4: jumlah digit nomor urut, yang akan di-padding dengan 0.

Contoh penggunaan:

```
$ ./nmass-renamer TEST TEST.OUT IMAGE 5
Copying TEST/gambar 1.png to TEST.OUT/IMAGE00001.png
Copying TEST/mobil keren (dapat dari teman).PNG to TEST.OUT/IMAGE00002.PNG
Copying TEST/new image 123.JPG to TEST.OUT/IMAGE00003.JPG
Copying TEST/pemandangan pantai 1.jPG to TEST.OUT/IMAGE00004.jPG
done
```

Contoh lain dengan prefix (file-), dan length (2) yang berbeda:

```
$ ./nmass-renamer TEST TEST.OUT.2 file- 2
Copying TEST/gambar 1.png to TEST.OUT.2/file-01.png
Copying TEST/mobil keren (dapat dari teman).PNG to TEST.OUT.2/file-02.PNG
Copying TEST/new image 123.JPG to TEST.OUT.2/file-03.JPG
Copying TEST/pemandangan pantai 1.jPG to TEST.OUT.2/file-04.jPG
done
```

Penjelasan source code:

- Pertama-tama, kita akan memeriksa jumlah argumen yang diberikan. Apabila tidak sama dengan 4, program akan keluar dengan *exit code* 1.

```
if [ $# -ne 4 ]
then
...
...
...
        exit 1
fi
```

- Kemudian, kita akan memeriksa apakah direktori output ditemukan. Apabila tidak, kita akan membuatnya.

```
if [ ! -d "$2" ]
then
        mkdir -p "$2"
fi
```

- *Counter* nama file diset ke 1.

```
COUNTER=1
```

- Untuk setiap file yang ditemukan
  - Kita akan membuat nama file baru, sesuai dengan prefix dan panjang nomor yang diminta. Untuk memformat, kita mempergunakan bantuan printf:

```
        NEW_FN=`printf "%s/%s%0$4d" $2 $3 $COUNTER`
```

  - Ekstensi file bisa kita dapatkan dengan cara berikut:

```
        NEW_EXT=`echo ${f##*.}`
```

  - Nama file baru kemudian didapatkan:

```
        NEW_FILE="$NEW_FN.$NEW_EXT"
```

  - Pengopian file ke file baru dilakukan:

```
        cp -a "$f" "$NEW_FILE"
```

  - Counter ditambahkan dengan 1:

```
        COUNTER=$(($COUNTER+1))
```

Beberapa catatan penting tentang program:

- Kita tidak mengubah sama sekali ekstensi file. Sebenarnya, kita bisa saja mengubahnya ke huruf besar atau huruf kecil (misalnya dengan bantuan program tr), namun karena nama file adalah *case sensitive* di Linux, kita tetap mempertahankan. Silakan disesuaikan dengan preferensi Anda.
- Kita tidak memindahkan file, namun mengopikan file. Anda bisa mengganti cp dengan mv, apabila ingin memindahkan file. Beberapa pengguna lebih senang memindahkan, terutama kalau memiliki koleksi yang memakan ruang harddisk yang besar. Ketika Anda melakukan pemindahan, pastikan Anda memberikan pemeriksaan tambahan, agar tidak ada proses yang dilakukan secara parsial.
- Dengan menggunakan dialog atau Xdialog, kita bisa menampilkan *progress bar* agar lebih menarik. Pembahasan tentang progress bar di shell script telah dibahas di *Infolinux*.

Demikianlah contoh program mass renamer kita. Silakan dimodifikasi sesuai dengan kebutuhan Anda. Selamat mencoba.

Noprianto [noprianto@infolinux.co.id]

System Call untuk Bekerja dengan Proses

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.

Proses secara sederhana dapat diartikan sebagai program yang sedang berjalan. Di sistem operasi, suatu proses memiliki status tertentu. Sebagai contoh, apakah suatu proses sedang berjalan atau tidak sedang berjalan. Kemudian, proses sendiri juga memiliki pemilik. Proses pun juga bisa membuka file, dan menariknya di Linux dan beberapa sistem operasi lain, proses pun bisa memiliki anak.

Cakupan pembahasan proses sendiri sangatlah luas. Satu buku pun rasanya tidak akan cukup untuk membahas penanganan proses di Linux. Kita, di tulisan ini, akan membahas sedikit tentang dasar-dasar proses. Di kesempatan-kesempatan lain, kita akan melangkah lebih lanjut mengeksplorasi penanganan proses di Linux.

Semua contoh di tulisan ini dibuat di atas Slackware 11, gcc 3.4.6 dan kernel 2.6.18, namun seharusnya dapat diterapkan tanpa masalah pada sistem lainnya.

Melihat daftar proses

Untuk melihat daftar proses, pergunakanlah program ps. Contoh:

```
$ ps
  PID TTY          TIME CMD
 3504 pts/0    00:00:00 bash
 3516 pts/0    00:00:00 ps
```

Pada contoh tersebut, kolom paling kiri adalah PID, atau *Process Identifier*. Setiap proses memiliki ID sendiri-sendiri yang unik.

Untuk melihat semua proses di sistem (sintaks BSD), berikanlah argumen ax ketika menjalankan ps:

```
$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init [3]
    2 ?        SN     0:00
[ksoftirqd/0]
    3 ?        S<     0:00
[events/0]
    4 ?        S<     0:00 [khelper]
    5 ?        S<     0:00 [kthread]
...
...
...
 3503 ?        Ss     0:00 rxvt
 3504 pts/0    Ss     0:00 bash
```

Untuk menampilkan daftar proses lengkap dengan *user* yang menjalankan (pemilik), berikanlah perintah berikut:

```
$ ps aux
...
...
nop       4020  0.1  0.2   3376
1844 ?        Ss   11:16   0:00 rxvt
nop       4021  0.0  0.2   3424
1968 pts/1    Ss   11:16   0:00 bash
```

PID lebih detail

PID sendiri merupakan integer positif mulai dari 1. Terakhir, penulis mendapatkan informasi, jumlah PID mampu mencapai angka 1 juta (kernel 2.6).

Proses dengan PID 1, yaitu init, merupakan nenek moyang proses. Program init sendiri merupakan program yang pertama dijalankan (setelah melalui initrd atau initramfs – apabila dipergunakan) pada saat sistem *boot*.

Untuk bekerja dengan proses, kita selalu bekerja dengan PID.

Tree proses

Proses bisa memiliki anak dan orang tua (kecuali init). Hubungan antara anak dan orang tua proses dari sisi internal kernel, merupakan hal yang cukup kompleks. Kita akan membahasnya pada kesempatan lain.

Untuk melihat *tree* proses dan mendapatkan gambaran besar hubungan anak dan orang tua, gunakanlah program pstree, seperti contoh berikut:

```
$ pstree
init-+-acpid
     |-6*[agetty]
     |-atd
     |-crond
     |-events/0
     |-inetd
     |-khelper
     |-klogd
     |-ksoftirqd/0
     |-kthread-+-aio/0
     |         |-ata/0
     |         |-ata_aux
     |         |-kacpid
     |         |-kblockd/0
     |         |-khubd
     |         |-kpsmoused
     |         |-kseriod
     |         |-kswapd0
     |         |-2*[pccardd]
     |         |-2*[pdflush]
     |         |-reiserfs/0
```

```
      |          |-xfsbufd
      |          |-xfsdatad/0
      |          |-xfslogd/0
      |          `-xfssyncd
      |-sshd
      |-syslogd
      |-udevd
...
...
...
(dipotong)
```

System call: mendapatkan PID

Untuk mendapatkan PID proses kita, gunakanlah system call getpid():

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

pid_t getpid(void);
```

Berikut ini adalah contoh penggunaan getpid(), yang disimpan pada source code s_getpid.c:

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

int main()
{
  pid_t pid;

  pid = getpid();

  fprintf (stdout, "PID saya adalah
%d\n", pid);

  return 0;

}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

```
$ gcc -o s_getpid s_getpid.c
```

Contoh output:

```
$ ./s_getpid
PID saya adalah 3716
```

Harap diperhatikan, program langsung selesai setelah menampilkan PID. Dengan demikian, PID tersebut tidak akan tampak pada keluaran program ps ax.

System call: mendapatkan PID proses orang tua

Untuk mendapatkan PID proses orang tua, gunakanlah system call getppid():

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

pid_t getppid(void);
```

Berikut ini adalah contoh penggunaan getppid(), yang disimpan pada *source code* s_getppid.c:

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

int main()
{
  pid_t pid, ppid;

  pid = getpid();
  ppid = getppid();

  fprintf (stdout, "PID saya adalah
%d\n", pid);
  fprintf (stdout, "PID proses orang
tua saya adalah %d\n", ppid);

  return 0;

}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

```
$ gcc -o s_getppid s_getppid.c
```

Contoh *output*:

```
$ ./s_getppid
PID saya adalah 3759
PID proses orang tua saya adalah
3504
```

System call: membatalkan proses

Ketika proses kita sedang berjalan, pada kondisi tertentu, kita bisa membatalkan program kita, seperti ketika dikenakan *signal* SIGABRT. Untuk membatalkan proses, gunakanlah system call abort():

```
#include <stdlib.h>

void abort(void);
```

Berikut ini adalah contoh penggunaan abort(), yang disimpan pada source code s_abort.c:

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

int main()
{

  abort();

  return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

```
$ gcc -o s_abort s_abort.c
```

Contoh output:

```
$ ./s_abort
Aborted
```

Sekilas tentang signal

Komunikasi proses satu dengan proses lain adalah topik yang sangat menarik. Dalam bentuk sangat sederhana, dan merupakan salah satu cara tertua, komunikasi proses dilakukan dalam bentuk saling mengirimkan signal. Ketika suatu proses menerima signal, terdapat tiga aksi yang bisa dilakukan:

- menuruti aksi *default* suatu signal. Misalnya, ketika suatu proses dikirimkan SIGTERM dengan aksi default proses diterminasi, maka proses akan diterminasi.
- Melakukan aksi sendiri. Berlawanan dengan aksi sebelumnya, proses dapat mengimplementasikan signal *handler* tertentu, untuk signal-signal yang dikirimkan kepadanya. Terdapat dua signal yang tidak bisa dibelokkan, yaitu SIGKILL dan SIGSTOP.
- Mengabaikan signal. Terdapat dua signal yang tidak bisa diblok, yaitu SIGKILL dan SIGSTOP.

Untuk mengirimkan signal ke suatu proses, kita bisa mempergunakan program *kill*. Untuk itu, kita perlu mengetahui PID proses dan signal yang ingin dikirim (signal default adalah SIGTERM).

Untuk mendapatkan daftar signal, cara termudah adalah dengan menjalankan program kill dengan argumen -l. Contoh:

```
$ kill -l
 1) SIGHUP       2) SIGINT
3) SIGQUIT      4) SIGILL
...
...
...
59) SIGRTMAX-5  60) SIGRTMAX-4
61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Lebih lanjut tentang penanganan signal, akan kita bahas pada kesempatan lain.

System call: Program bunuh diri

Proses yang bunuh diri, tentu tidak ada hubungannya dengan fenomena bunuh diri

yang cukup sering terjadi di masyarakat. Di contoh ini, kita akan membahas program yang melakukan bunuh diri, setelah mencapai kondisi tertentu.

Secara teknis, bunuh diri berarti mengirimkan signal yang mematikan kepada diri sendiri. Untuk mengirimkan signal, kita mempergunakan system call kill():

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

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

Berikut ini adalah contoh penggunaan kill(), yang disimpan pada source code s_kill_1.c:

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

int main()
{

  pid_t pid;
  int i;


  pid = getpid();

  fprintf (stdout, "Self terminating program\n");
  fprintf (stdout, "My PID is %d\n", pid);

  for (i=5; i>=1; i--)
  {
   fprintf (stdout, "%10i\n", i);
   sleep (1);
  }

  fprintf (stdout, "Sending my self the TERM signal\n");

  kill (pid, SIGTERM);

  while (1)
  {
   fprintf (stdout, "Should not reach here\n");
  }

  return 0;

}
```

Lakukanlah kompilasi, dengan memberikan perintah berikut:

```
$ gcc -o s_kill_1 s_kill_1.c
```

Contoh output:

```
$ ./s_kill_1
Self terminating program
My PID is 3957
          5
          4
          3
          2
          1
Sending my self the TERM signal
Terminated
```

Penjelasan source code:

- Pertama-tama, kita mendapatkan PID proses dengan getpid().
- Kita melakukan perulangan yang mencetak 5,4,3,2,1, di mana masing-masing perulangan ditunda selama 1 detik.
- Setelah perulangan selesai, kita mengirimkan SIGTERM kepada diri sendiri. Ini kita lakukan dengan system call kill(). Proses harusnya diterminasi.
- Apabila proses gagal diterminasi, maka program akan secara terus menerus menampilkan tulisan *Should not reach here*.
- Catatan tambahan: untuk mengirimkan signal ke diri sendiri, kita bisa pula mempergunakan system call raise().

System call: membuat program kill sederhana

Di contoh ini, kita akan membuat sendiri program kill, mempergunakan system call kill() yang telah dibahas sebelumnya. Berikut ini adalah isi file s_kill_2.c:

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

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

  pid_t pid;
  int sig;
  int ret;

  if (argc != 3)
  {
   fprintf (stderr, "Usage %s <pid> <signal_number>\n", argv[0]);
   exit (1);
  }

  pid = atoi (argv[1]);
  sig = atoi (argv[2]);

  ret = kill (pid, sig);
  if (ret != 0)
  {
   fprintf (stderr, "%s\n", strerror(errno));
   exit (errno);
  }
  else
  {

   fprintf (stdout, "%s\n", strerror(errno));
  }

  return 0;

}
```

Lakukanlah kompilasi, dengan memberikan perintah berikut:

```
$ gcc -o s_kill_2 s_kill_2.c
```

Contoh output:

```
$ ./s_kill_2
Usage ./s_kill_2 <pid> <signal_number>

$ ./s_kill_2 1 9
Operation not permitted
```

Pada contoh ini, sebagai user biasa,kita ingin 'membunuh' init (PID 1) dengan signal SIGKILL. Tentu saja tidak diizinkan.

```
$  ./s_kill_2 3999 9
Success
[1]+  Killed                  vi
```

Pada contoh ini, kita menjalankan vi, kemudian menekan *CTRL-Z* untuk menjadikannya proses *background*.

```
$ ./s_kill_2 4007 15
Success
```

Ketika kita ingin menjalankan kembali vi di *foreground*, vi akan menampilkan pesan berikut:

```
Vim: Caught deadly signal TERM
Vim: Finished.
Terminated
```

Sampai di sini dulu pembahasan kita. Di berbagai kesempatan lain, kita akan membahas lagi penanganan proses di Linux. Selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

Membuat Video CD dengan Mudah

Video CD merupakan media distribusi film yang populer. Di tulisan singkat ini, kita akan membahas pembuatan video CD memanfaatkan beragam format video, yang umum ditemukan di komputer. Pembuatan Video CD di Linux dapat dilakukan dengan mudah dan cepat.

Membuat video CD di Linux bisa dilakukan dengan berbagai cara. *Software* yang dapat digunakan juga sangat beragam. Mulai dari *suite* aplikasi yang komplit, sampai *tool* kecil-kecil yang memiliki fungsi spesifik.

Di tulisan ini, kita akan membuat video CD memanfaatkan beberapa tool *command line*. Distro yang dipergunakan di dalam tulisan ini adalah Slackware 11, namun seharusnya bisa diterapkan pada distro lain tanpa masalah.

Secara umum, pembuatan video CD bisa dilakukan dalam beberapa tahap berikut:

- Mengonversi format video ke format Video CD. Seperti kita ketahui bersama, format video yang digunakan di video CD adalah MPEG-1.
- Membuat *image* VCD dari file-file video yang telah dipersiapkan sebelumnya.
- Membakar image VCD yang dihasilkan sebelumnya.

1. Konversi format video

Untuk mengonversi format video ke MPEG-1, kita bisa mempergunakan program ffmpeg, yang bisa didapatkan dari *http://ffmpeg.org*. Bacalah file INSTALL yang terdapat di dalam *archive* untuk langkah instalasi.

Penulis mempergunakan ffmpeg versi SVN-r8757. Apabila Anda mempergunakan versi berbeda, apabila diperlukan, sesuaikanlah parameter yang digunakan.

Di dalam contoh konversi ini, kita akan mengonversi file video yang diekstrak dari VCD, menggunakan program Windows Explorer di Windows. Nama file video tersebut, umumnya memiliki ekstensi .DAT.

Menggunakan program file, kita bisa mengetahui informasi file .DAT tersebut:

```
$ file temp/movie/ICE_AGE_2/1/
AVSEQ04.DAT
temp/movie/ICE_AGE_2/1/AVSEQ04.DAT:
RIFF (little-endian) data, wrapped
MPEG-1 (CDXA)
```

Sementara, kita membutuhkan file dengan format MPEG-1.

Untuk setiap file video yang ingin dimasukkan ke Video CD, lakukanlah konversi dengan menjalankan perintah berikut:

```
$ ffmpeg -i <FILE_VIDEO_YANG_INGIN_
DIKONVERSI> -target pal-vcd <FILE_
VIDEO_OUTPUT.mpg>
```

Contoh:

```
$ ffmpeg -i ~/temp/movie/ICE_AGE_
2/1/AVSEQ04.DAT -target pal-vcd
1.mpg
FFmpeg version SVN-r8757, Copyright
(c) 2000-2007 Fabrice Bellard, et
al.
  configuration: --prefix=/usr --
enable-shared --enable-gpl --enable-
pp --enable-pthreads --disable-debug
--enable-liba52 --enable-libfaac --
enable-libfaad --enable-libmp3lame
--enable-libogg --enable-libtheora
--enable-libvorbis --enable-x264 --
enable-xvid
  libavutil version: 49.4.0
  libavcodec version: 51.40.4
  libavformat version: 51.12.1
  built on Apr 19 2007 15:14:13,
gcc: 3.4.6
Input #0, mpeg, from '/home/nop/
temp/movie/ICE_AGE_2/1/AVSEQ04.DAT':
  Duration: 00:46:40.4, start:
0.826667, bitrate: 1411 kb/s
  Stream #0.0[0x1c0]: Audio: mp2,
44100 Hz, stereo, 224 kb/s
  Stream #0.1[0x1e0]: Video:
mpeg1video, yuv420p, 352x288, 1140
kb/s, 25.00 fps(r)
Output #0, vcd, to '1.mpg':
  Stream #0.0: Video: mpeg1video,
yuv420p, 352x288, q=2-31, 1150 kb/s,
25.00 fps(c)
  Stream #0.1: Audio: mp2, 44100 Hz,
stereo, 224 kb/s
Stream mapping:
  Stream #0.1 -> #0.0
  Stream #0.0 -> #0.1
Press [q] to stop encoding
...
...
...
frame=70040 fps= 67 q=2.0
Lsize=  475517kB time=2801.6
bitrate=1390.5kbits/s
video:393281kB audio:76607kB
global headers:0kB muxing overhead
1.197852%
```

Pada contoh ini, sebuah file 1.mpg akan dihasilkan. File 1.mpg tersebut telah bertipe:

```
$ file 1.mpg
```

```
1.mpg: MPEG sequence, v1, system
multiplex
```

2. Membuat image VCD

Sampai di tahap ini, aturlah semua file video yang telah dikonversi di tahap sebelumnya, yang ingin dimasukkan ke dalam setiap keping VCD. Satu keping VCD tentunya bisa berisikan beberapa *track*, yang setiap track-nya diwakili oleh sebuah file video.

Untuk membuat image VCD, kita akan mempergunakan program vcdimager, yang bisa di-*download* di *http://www.gnu.org/software/vcdimager/*. Bacalah file INSTALL yang terdapat di dalam archive, untuk langkah instalasi.

Versi vcdimager yang penulis gunakan adalah 0.7.23.

Untuk setiap keping VCD, lakukanlah pembuatan image, dengan memberikan perintah berikut:

```
$ vcdimager <TRACK1> [TRACK2 ..
TRACKN]
```

Contoh:

```
$ vcdimager  1.mpg
++ WARN: initializing libvcd 0.7.23
[linux-gnu/i486]
++ WARN:
++ WARN:  this is the Beta
development branch!
++ WARN:  use only if you know what
you are doing
++ WARN:  see http://www.hvrlab.
org/~hvr/vcdimager/ for more
information
++ WARN:
finished ok, image created with
210197 sectors [46:42:47] (494383344
bytes)
```

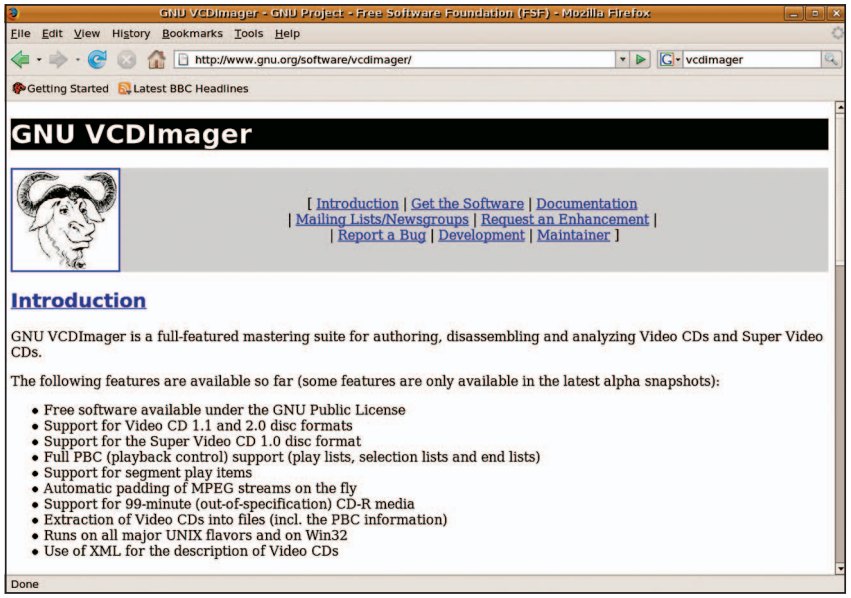
Perintah tersebut akan menghasilkan file dengan nama *default*, videocd.bin dan videocd.cue. Berikut ini adalah tipe masing-masing file tersebut:

```
$ file videocd.bin
videocd.bin: data

$ file videocd.cue
videocd.cue: ASCII text, with CRLF
line terminators
```

Untuk nama file *output* non-default (videocd.bin dan videocd.cue), gunakanlah opsi:

- -c: file cue
- -b: file bin



Situs web vcdimager.

3. Membakar Image Video CD

Untuk membakar image video CD yang dihasilkan sebelumnya, kita akan mempergunakan program cdrdao, yang bisa di-download di *http://cdrdao.sourceforge.net*. Bacalah file INSTALL yang terdapat di dalam archive, untuk langkah instalasi.

Versi cdrdao yang penulis gunakan adalah 1.2.1.

Masukkanlah CD kosong ke dalam *drive*-nya, kemudian, untuk setiap file .cue yang dihasilkan pada langkah sebelumnya, lakukanlah pembakaran dengan perintah berikut (sebagai *root*):

```
# cdrdao write -device <DEV> --speed
<SPEED> <FILE_CUE>
```

Contoh:

```
# cdrdao write --device 0,0,0 --
speed 16 videocd.cue
Cdrdao version 1.2.1 - (C) Andreas
Mueller <andreas@daneb.de>
...
...
Using libscg version 'schily-0.8'

0,0,0: LITE-ON COMBO SOHC-5236V Rev:
R$0C
Using driver: Generic SCSI-3/MMC -
Version 2.0 (options 0x0000)

Starting write at speed 16...
Pausing 10 seconds - hit CTRL-C to
abort.
Process can be aborted with QUIT
signal (usually CTRL-\).
Turning BURN-Proof on
Enabling JustSpeed.
ERROR: Cannot set Ricoh mode page 30.
Executing power calibration...
Power calibration successful.
Writing track 01 (mode MODE2_RAW/
MODE2_RAW )...
Writing track 02 (mode MODE2_RAW/
MODE2_RAW )...
Wrote 471 of 471 MB (Buffers 100%
96%).
Wrote 210197 blocks. Buffer fill min
87%/max 100%.
Flushing cache...
Writing finished successfully.
```

Untuk mendapatkan <DEV>, gunakanlah perintah berikut:

```
# cdrdao scanbus
```

Contoh:

```
# cdrdao scanbus
Cdrdao version 1.2.1 - (C) Andreas
Mueller <andreas@daneb.de>
...
...
Using libscg version 'schily-0.8'

0,0,0 : LITE-ON , COMBO SOHC-5236V,
R$0C
ATA:1,0,0             MITSUMI ,
CD-ROM SR244W1  , T01A
```

Demikianlah tiga langkah mudah membuat Video CD di Linux. Sampai di sini dulu pembahasan kita. Selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

Mengatur Permission di Unix

Hak akses atau *permission* di sistem Unix/Linux, merupakan suatu konsep kepemilikan file/folder yang penting untuk dipahami oleh para pengguna Linux. Dalam tutorial ini, akan dijelaskan dasar-dasar permission di Unix/Linux, berikut dengan contoh kasus dan implementasi dari konsep permission ini.

Seorang pemula Unix/Linux, umumnya cepat atau lambat akan harus mengenali dan memahami permission *filesystem* di Unix. Baik lewat perintah *command-line* populer "ls" ataupun lewat file *manager* berbasis GUI. Dalam waktu singkat, ia akan mengamati adanya konsep "owner", "group", maupun sederetan bit permission, seperti "drwxr-xr-x". Artikel ini akan menjelaskan dasar-dasar permission di Unix, serta contoh-contoh nyata pengaturan permission untuk mengimplementasi berbagai *policy* keamanan. Sebagian contoh diambil dari skenario *server shared hosting*, yang merupakan contoh yang baik, untuk melihat bagaimana mengatur permission di lingkungan *multiple user*.

Contents:

1. Konsep dasar

Setiap file/direktori di filesystem memiliki tiga atribut: owner, group, dan mode (atau *protection*, atau kadang disebut permission juga). Ketiganya berupa angka. Untuk menyatakan permission secara lengkap, notasinya biasanya (owner, group, mode).

Owner adalah Unix uid (bertipe uid_t, umumnya 32-bit). Biasanya oleh "ls" atau file manager yang ditampilkan adalah *username-nya* (kecuali jika username-nya tidak ada; ini mungkin terjadi jika kita mengekstrak atau mengopi file dari sistem lain, di mana daftar user-nya tidaklah sama).

Group adalah Unix gid (bertipe gid_t, umumnya 32-bit). Sama seperti owner, biasanya oleh "ls" atau file manager yang ditampilkan adalah *group name*-nya, agar lebih manusiawi.

Mode (atau permission) adalah sederetan bit 0 dan 1 untuk menyatakan *flag* akses (bertipe mode_t, umumnya berupa angka 32-bit). Ada 9-bit permission dasar, yaitu r, w, x untuk owner, rwx untuk group, dan rwx untuk other (atau world). Flag r (*read*) mengatur apakah sebuah file dapat dibaca. Flag w (*write*) mengatur apakah sebuah file dapat ditulisi. Flag x (*execute*) mengatur apakah sebuah objek dapat dijalankan (di-"*run*").

Untuk direktori, bit r artinya sebuah direktori dapat dilihat isinya (dapat di-"ls"), dan bit x artinya direktori tersebut dapat dimasuki (dapat di-"cd"). Bit w artinya kita dapat memasukkan file atau menghapus file di direktori tersebut.

Dengan huruf-huruf rwx, permission ditulis dengan urutan owner, group, dan other. Contoh: rwxr-x--- artinya semua bit rwx menyala untuk owner (rwx), hanya bit r dan x menyala untuk group (r-x), dan tidak satupun rwx menyala untuk *other* (---).

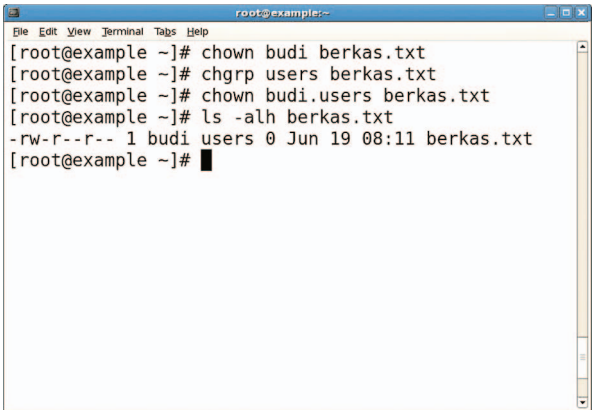
Dengan bilangan basis 8 (oktal), permission ini dapat diekspresikan dengan tiga digit. Bit r bernilai 4, w bernilai 2, x bernilai 1. Contohnya adalah 755 (kadang harus ditulis 0755, di mana prefiks "0" menandai bahwa ini oktal). Permission 755 artinya semua bit rwx menyala untuk owner (rwx), r dan x untuk group (r-x), dan r serta x untuk other (r-x). Di contoh sebelumnya, rwxr-x--- ekivalen dengan 750.

Selain 9-bit permission dasar ini, ada juga beberapa bit lain seperti setuid, setgid, sticky, dll. Sebagian dari bit-bit ini akan dijelaskan sambil jalan.

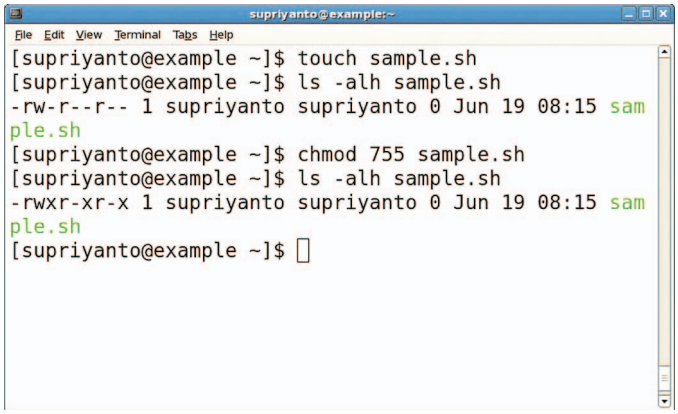
Perlu dicatat juga bahwa *user root* mem*bypass* hampir semua pengecekan ini. Yang mana menjadi salah satu alasan untuk menghindari menggunakan root untuk kegiatan sehari-hari, karena file dan direktori kita dapat tertimpa/terhapus oleh root, walaupun permissionnya sudah diset cukup restriktif.

2. Mengeset ownership

Untuk mengeset atau mengubah ownership di command line Unix, digunakan perintah *chown* ("change owner") dan *chgrp* ("change group"). Contoh, untuk mengubah file berkas.txt menjadi milik budi:



Mengubah hak kepemilikan file berkas.txt.



Mengubah hak kepemilikan file sample.sh.

```
# chown budi berkas.txt
```

Untuk mengubah group menjadi group users:

```
# chgrp users berkas.txt
```

Chown memiliki sintaks *shortcut* untuk melakukan pengubahan owner dan group:

```
# chown budi.users berkas.txt
```

Chown hanya dapat dilakukan oleh root. Chgrp dapat dilakukan oleh root atau oleh user yang memiliki file/direktori. Jika dilakukan oleh si pemilik file yang bukan root, maka pengubahan group hanya dapat dilakukan ke salah satu group di mana si user tersebut adalah *member*. Pada contoh chgrp di atas, jika perintah dieksekusi oleh user Budi, maka si Budi haruslah memberi dari group users. User dilarang mengubah file-nya ke group asing lain.

3. Mengeset permission

Untuk mengeset permission di command line Unix, digunakan perintah *chmod* ("change mode"). Contoh:

```
$ chmod  755 NAMAFILE
$ chmod 0755 NAMAFILE ; # sama saja
```

Perintah chmod juga mengizinkan kita menghidupmatikan bit secara fleksibel. Contoh untuk mematikan semua bit x (execute):

```
$ chmod -x NAMAFILE
```

Untuk menghidupkan bit w dan x pada other:

```
$ chmod o+rx NAMAFILE
```

Chmod hanya dapat dilakukan oleh root atau oleh si pemilik file.

4. Pengaturan publik dan privat

Rata-rata file di Unix, umumnya bersifat *publicly readable*: "dapat dibaca/dieksekusi oleh siapa saja, tapi hanya bisa dimodifikasi oleh *admin*". Permission-nya adalah (root, root, 644) atau (root, root, 755) untuk program. Contohnya amat banyak, misalnya semua file di */usr/bin*, */usr/share*, dll. Semua dokumentasi, program, *library*, umumnya bersifat publik di Unix. *Toh*, semua file-file ini memang tidak berisi data rahasia yang harus ditutup-tutupi.

Bahkan lokasi-lokasi tertentu sifatnya lebih bebas lagi, *publicly writable* (atau *world writable*): *free for all*. Dapat ditulisi oleh siapa saja. Contoh yang paling terkenal adalah */tmp*. Permission direktori ini adalah (root, root, 777). Setiap orang dapat menaruh file di sini karena bit w pada kolom *other* menyala. Artinya setiap orang memiliki akses "write" pada direktori ini.

Tapi, agar tidak terlalu "anarkis", maka ada bit permission tambahan lain yang dihidupkan pada direktori /tmp ini, yaitu *bit sticky*. Dengan hidupnya bit ini, maka ada restriksi tambahan: file di dalam direktori hanya boleh diganti nama (di-*rename*) atau dihapus oleh pemiliknya sendiri. Artinya, Anda bisa saja menambah file sesuka hati ke /tmp, tapi tidak bisa seenaknya menghapus atau mengganti nama file milik orang lain.

Umumnya, direktori-direktori free for all ditambahi bit sticky ini. Untuk menghidupkan bit sticky di command line, dapat digunakan perintah "chmod +t" atau mode numerik 1000, contoh: "chmod 1777". Di perintah "ls", Anda akan melihat huruf "t" atau "T" di ujung deretan permission mode sebagai tanda bahwa bit sticky menyala.

Oke, rata-rata file di sistem bersifat publik. Namun tentu ada pula yang bersifat privat: hanya dapat dibaca oleh user tertentu saja (atau bahkan hanya oleh root saja).

Contohnya direktori home user (*/home/USER*) diset (*USER, USER, 700*). Permission seperti ini berarti, hanya user yang bersangkutan saja yang dapat masuk. Catatan: ada juga distro-distro Linux yang secara *default* membuka direktori home ini menjadi 755 atau 750, jadi Anda sebagai user perlu berhati-hati, dan memproteksi lagi file-file penting dan privat di dalam home Anda dengan 600/700.

Contoh lain file privat adalah file berisi *password* sistem (/etc/shadow) yang diset (root, root, 600). Hanya root saja yang dapat melihat, dan menulis isi file ini.

Contoh lain lagi, swap file. Karena dapat berisi segala macam informasi sistem yang sensitif, maka harus diset setidaknya 600.

5. Pengaturan berbasis group

Untuk pengaturan yang lebih fleksibel dan di tengah-tengah spektrum ("tidak benar-benar publik, tapi tidak benar-benar privat"), digunakan pengaturan berbasis group. Seperti diketahui, sebuah group dapat beranggotakan lebih dari satu user, dan sebuah user dapat menjadi anggota lebih dari satu group. Dengan cara ini, kita dapat memberi atau mematikan akses pada sekelompok user tertentu.

Kita dapat membuat group-group baru dengan perintah *groupadd*:

```
# groupadd group1
```

Lalu menambahkan user ke dalam group ini:

```
# gpasswd -a budi group1
```

Untuk mengeluarkan sebuah user dari keanggotaan sebuah group:

```
# gpasswd -d budi group1
```

Jika sebuah group sudah tidak dibutuhkan, dapat dihapus:

```
# groupdel group1
```

6. Pengaturan group: inklusi

Beberapa file atau lokasi perlu dapat diakses oleh sekelompok user saja. Untuk ini, kita dapat mengeset file/direktori tersebut dengan permission (root/USER, GROUP, 640/660/750/770). Dengan mematikan semua bit rwx pada kolom other, maka di luar anggota GROUP tidak ada yang dapat mengakses.

Salah satu contoh adalah file-file *log* di bawah /var/log (seperti log webserver /var/log/apache2 atau log FTP server /var/log/xferlog), yang hanya diperuntukkan oleh para *administrator* (group adm). User biasa tidak diperbolehkan mengintip, karena log-log ini biasanya berisi catatan untuk semua user. Yang berarti dapat membocorkan lokasi file user-user lain.

Sebuah contoh lain, di *server hosting* direktori /u/USER/sites diset (root, USER, 751). Direktori ini berisi file-file *website*, masing-masing website dalam subdirektori tersendiri (contoh: /u/USER/sites/site1.com, /u/USER/sites/site2.org, dst). Selain si USER, user lain termasuk user www-data (*web server*) juga dapat masuk (bit x), tapi tidak dapat melihat isi website. Dengan begini, user lain tidak dapat mengetahui (me-*list*) di bawah /u/USER/sites ada website apa saja. Nama masih dapat masuk lebih dalam, jika ia sudah mengetahui nama persis website tersebut.

Contoh lain lagi, di server hosting direktori /proc diset menjadi (root, support-staff, 0550). Dengan demikian user yang bukan anggota *support-staff* tidak dapat mengakses /proc. Perintah-perintah seperti ps, top, netstat yang dijalankan non-staff menjadi tidak berfungsi. Ini berguna untuk menjaga privasi sistem, agar user tidak dapat mengintip terlalu banyak kondisi sistem, karena /proc banyak berisi informasi sistem termasuk proses-proses user lain.

Catatan: Karena permission /proc ini akan te-*reset* setiap kali *boot* saat *mount*, maka kita perlu mengesetnya lagi setelah proses mount.

7. Pengaturan group: eksklusi

Kadang-kadang yang kita ingin lakukan justru sebaliknya, mengunci akses dari sekelompok user saja. Biasanya, di server hosting ini dilakukan untuk menutup akses dari "user-user nakal". Contoh: /usr/sbin/sendmail dapat diset menjadi (root, no-sendmail, 0705). User-user yang nakal karena sering mengirim *junk e-mail*, dapat dimasukkan sementara ke dalam group no-sendmail, sehingga tidak dapat mengakses program /usr/sbin/sendmail. Jika sudah bertobat dan mengaku dosa, dapat dikeluarkan lagi dari group no-sendmail.

Contoh lain, /var/log diset (root, hosting-user, 0705). Semua user hosting (yang dimasukkan ke group hosting-user, tentunya) tidak diperbolehkan masuk melihat-lihat.

Contoh lain lagi, /bin/su dapat diset (root, no-su, 4705). Artinya semua user, kecuali yang masuk ke dalam group no-su, diperbolehkan mengakses program ini.

Apa artinya angka 4? Ini adalah bit permission mode lain yang dinamakan setuid. Mode ini memiliki arti spesial, yaitu sebuah file setuid jika dijalankan akan berjalan bukan sebagai user yang memanggilnya, tapi sebagai user owner file. Karena owner program /bin/su adalah root, maka tak peduli siapapun yang memanggil program ini, maka su akan berjalan sebagai root. Seperti kita tahu, su ("switch user") berguna untuk berganti user. Kemampuan ajaib berganti-ganti user ini baru dapat diperoleh, jika berjalan sebagai root. Beberapa program lain juga berjalan sebagai setuid-ke-root, contohnya /usr/bin/passwd untuk mengganti password (karena harus bisa menulisi /etc/shadow). Membuat program setuid harus berhati-hati, karena dapat berisiko memberi akses pada user yang tidak seharusnya. Karenanya menutup akses program-program ini dengan mode permission yang lebih restriktif, ada bagusnya juga.

Perlu dicatat bahwa memprotek program non-setuid dengan proteksi user (mis: 700) atau proteksi group (mis: 750 atau 705), sebetulnya untuk tujuan pencegahan eksekusi tidaklah benar-benar efektif, karena orang dapat saja meng-*upload* binari sendiri yang serupa. Tapi memproteksi program setuid dapat efektif, karena bit setuid tidak dapat seenaknya diset sendiri.

8. Pengaturan group: multiple group logika AND

Bagian ini hanya untuk pengguna mahir. Dengan trik yang penulis namakan direktori pembungkus, kita dapat mengimplementasikan kebijakan keamanan menutup/membuka akses, terhadap dua atau lebih group sekaligus. Konsepnya adalah karena sebuah file/direktori hanya dapat diberi atribut 1 group saja, maka kita menaruh file/direktori yang ingin kita protek tersebut di bawah satu atau lebih subdirektori lain, dan mengenakan atribut group-group tambahannya pada si subdirektori pembungkus tadi. Ibaratnya menaruh pintu berlapis, seperti boneka bersusun Rusia.

Contoh, untuk dua group:

```
# mengizinkan target diakses oleh
anggota group A dan B:
dir/       = (x, A, 750)
dir/target = (x, B, 750)
```

Efek perintah di atas adalah seorang user harus menjadi anggota kedua group dulu, sebelum bisa masuk.

```
# menutup akses dari anggota group
A dan B:
dir/       = (x, A, 705)
dir/target = (x, B, 705)
```

Efek perintah di atas adalah jika seorang user anggota salah satu group, maka tidak bisa masuk ke target. Entah terblokir di dir/ (karena anggota group A) atau di dir/target (karena anggota group B).

Untuk tiga group atau lebih, cukup perdalam levelnya.

9. Pengaturan group: multiple group logika OR

Bagian ini hanya untuk pengguna mahir. Untuk logika OR, kita dapat menggunakan fasilitas *link* di Unix. Contoh, untuk dua group:

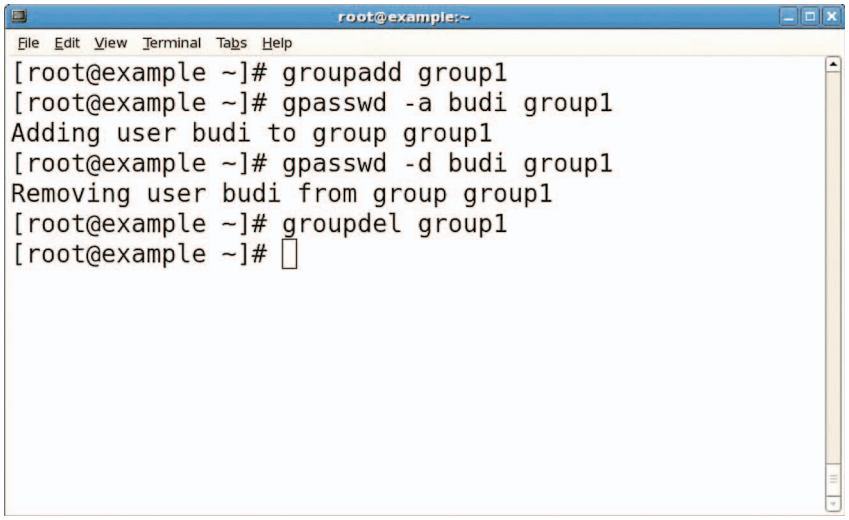
```
# mengizinkan target diakses oleh
anggota group A atau B:
dir1   (x, A, 750)
dir1/target
dir2   (x, B, 750)
dir2/target adalah link dari dir1/
target
```

Efek dari perintah di atas, user anggota A dapat mengakses target lewat dir1. Anggota B dapat mengakses *via* dir2.

Untuk tiga group atau lebih, cukup memperbanyak "gerbang alternatif"-nya.

10. Beberapa contoh lain

Beberapa contoh lain pengaturan permission di Unix.



Cara membuat dan menghapus group, serta menambah dan men-delete user group.

Direktori session: /var/lib/php5 = (root, root, 1633). Direktori ini dipakai untuk menyimpan file sementara, seperti session. Semua user dapat menaruh file di sini. Tapi karena session ID (nama file) perlu dirahasiakan dari user lain, maka bit r dimatikan pada kolom other. Artinya, direktori ini tidak bisa di-"ls".

Mengunci file: (root, root, 0). Dengan mematikan semua bit, maka user selain root tidak dapat lagi mengakses file ini. Dengan tampilan "ls" pun jadi terindikasi jelas secara visual, bahwa file ini di-*disable* (mode-nya terlihat "---------").

File backup: chmod -w (400/440/444). Sebagian editor sengaja mematikan permission w pada file backup. Tujuannya adalah agar file tersebut tidak bisa diedit. Tentu saja ini bukan proteksi menyeluruh, hanya untuk membantu saja. User masih bisa tak sengaja menghapus file tersebut, dan jika ingin mengedit, bisa saja melakukan *chmod +w* dulu pada file tersebut.

11. Sebuah contoh lengkap

Berikut ini sebuah contoh yang agak lengkap, hierarki direktori untuk sebuah *account* di server shared hosting beserta penjelasannya.

```
/u/steven/          (root, steven, 755)
/u/steven/home/     (steven, steven, 700)
/u/steven/public/   (steven, steven, 755)
/u/steven/mysql/    (mysql, steven, 550)
/u/steven/mysql/db1/        (mysql, steven, 2750)
/u/steven/sites/    (root, steven, 751)
/u/steven/sites/site1.com/  (www-data, steven, 550)
/u/steven/sites/site1.com/www  (steven, steven, 755)
/u/steven/sites/site1.com/syslog/  (root, steven, 750)
```

Direktori account teratas (/u/steven) terbuka untuk siapa saja, karena di dalamnya berbagai user sistem (mis: apache, mysql) harus dapat masuk. Owner diset menjadi root, agar user steven tidak dapat membuat sembarang direktori di dalam direktori ini.

Direktori home (/u/steven/home) menjadi daerah privat milik user sepenuhnya. Tidak boleh ada user lain yang dapat masuk.

Direktori mysql (/u/steven/mysql) diset demikian, agar hanya mysql (dan user) yang dapat masuk. Direktori database (/u/steven/mysql/db1) diset menjadi milik mysql, agar hanya mysql yang dapat membuat file-file tabel dll. Namun diset setgid (2000), yang artinya file-file yang tercipta di bawah direktori tersebut otomatis group-nya diset menjadi user steven. Ini berguna untuk tujuan perhitungan group *quota*.

Direktori sites (/u/steven/sites) perlu dapat dimasuki oleh user itu sendiri, oleh apache, dan oleh beberapa user sistem lain. Tapi user sistem lain atau user hosting lain tidak dapat mengintip nama-nama situs di bawah *sites/*.

12. Permission tradisional vs ACL

Model permission yang telah dijelaskan di artikel ini, sebetulnya disebut model tradisional Unix (atau model tradisional POSIX). Mengapa disebut tradisional? Karena telah ada sejak lama, sejak zaman-zaman awal Unix dikembangkan, bahkan jauh sebelum Linux lahir. Praktis semua varian Unix mendukung model ini. Alasan lain disebut tradisional adalah karena telah dikembangkan model permission yang lebih fleksibel, seperti ACL (*access control lists*).

Dengan ACL, pengaturan permission bisa lebih fleksibel. Konsepnya adalah setiap file/direktori dapat memiliki daftar aturan akses. Dengan aturan yang dapat lebih dari satu inilah diperoleh fleksibilitas. User-user dan group-group yang berbeda dapat diberi aturan akses yang berbeda-beda. Misalnya, untuk sebuah file A: user A, B, C dapat membaca dan menulis. Group D, E juga dapat menulis, tapi group F sama sekali tidak dapat mengakses, dst. Dengan model tradisional POSIX, hal ini sulit dilakukan. Bisa, tapi akan melibatkan penciptaan group-group baru, karena sebuah file/direktori hanya bisa diasosiasikan dengan satu group saja.

Tapi kenapa sampai hari ini, model tradisional POSIX tetap mendominasi dan populer?

Alasan utama adalah karena kesederhanaannya. Karena sederhana, mudah dimengerti. Karena sederhana, mudah ditampilkan secara ringkas. Karena sederhana, mudah diuji/diaudit. Alasan lain adalah karena hingga hari ini masih ada tool-tool filesystem yang belum *aware* terhadap keberadaan ACL. Sehingga jika tidak hati-hati, ACL ini dapat hilang/tertinggal saat file dipindah-pindah, dan akibatnya tentu risiko keamanan. Di lain pihak, praktis semua tool Unix mendukung model tradisional POSIX, dari tool sederhana, seperti cp, rsync, tar hingga tool GUI dan komersial.

13. Penutup

Model permission tradisional UNIX/POSIX adalah salah satu dasar terpenting keamanan di Unix/Linux. Pastikan Anda benar-benar familiar, dan memahami konsep permission ini.

Steven Haryanto [steven@masterwebnet.com]

Lebih Lanjut dengan Lokasi Instalasi Program

Kompilasi dan instalasi program di Linux selalu menjadi topik yang menarik untuk dibicarakan. Ribuan *developer* membangun program dengan beragam pemaketan, dan instruksi untuk kompilasi. Ratusan distro dan segala perbedaannya menambah kompleksitas tersebut.

Kadang-kadang, kita mungkin pernah berpikir: alangkah sederhananya kalau hanya ada satu distribusi Linux. Satu cara konfigurasi sistem. Satu *package management system*. Dengan demikian, paling tidak instalasi program juga datang dengan cara yang serupa. Yang walaupun sulit, namun lama kelamaan, bisa dikuasai.

Sayangnya, dunia Linux saat ini tidaklah demikian. Barangkali, kondisi tidak demikianlah yang membantu Linux bisa semaju seperti saat ini.

Khusus untuk masalah instalasi program saja, dengan banyaknya distro Linux dan package management-nya, kita setidaknya bisa melakukan tiga hal berikut:

- Menuruti package management distro. Ini merupakan cara teraman, walaupun memiliki kompleksitas tersendiri.
- Melakukan kompilasi dari *source code*. Kita akan membahasnya di tulisan ini.
- Menggunakan package management lain.

Khusus untuk kompilasi program dari source code, kita pun masih harus berhadapan dengan sejumlah isu, di antaranya:

- Sistem yang digunakan dalam pemaketan, dan konfigurasi source code.
- Pustaka yang dibutuhkan, dan segala *dependency*.
- Integrasi dengan sistem.
- *Upgrade*.

Apabila semua hal tersebut ingin dibahas, maka tulisan ini jelas tidak akan cukup. Sebuah buku mungkin akan dibutuhkan. Oleh karena itu, kita akan melakukan pembahasan kompilasi dari source code, dengan memberikan sejumlah asumsi terlebih dahulu:

- Ada kemungkinan hasil kompilasi akan dipaketkan sesuai package management distro kita. Hal ini di antaranya akan menyelesaikan masalah upgrade, integrasi sistem, dan masalah dependency. Pembahasan tentang pembuatan paket distribusi sudah pernah dilakukan di edisi-edisi sebelumnya.
- Sistem telah terinstal semua pustaka dan *tool,* yang dilakukan untuk kompilasi program.
- Semua source code dipaketkan dengan bantuan autoconf, di mana umumnya untuk melakukan instalasi, kita cukup menjalankan serangkaian perintah *configure, make* dan *make install*.
- Saat ini, kita abaikan dulu beberapa standar yang ingin diusahakan oleh komunitas Linux, khususnya yang mengatur bagaimana *binary* harus ditempatkan.

Barulah, kita akan masuk ke topik tulisan kita: bagaimana saya mengatur lokasi instalasi program? Terlepas dari standar. Apakah:

- Kita kumpulkan semua di /opt?
- Kita tempatkan semua program/pustaka yang kurang penting di *prefix /usr/local/*.
- Kita *customize* lagi dengan detail semua lokasi binary, konfigurasi, *share* data, dan lain sebagainya.

Semua pembahasan di tulisan ini dilakukan di atas sistem Slackware 11, namun dapat diterapkan pada sistem lain tanpa masalah. Untuk program contoh, kita akan mempergunakan moc, sebuah *console* audio player untuk Linux, yang bisa didapatkan dari *http://moc.daper.net*. Versi moc yang digunakan adalah 2.4.1.

Sebelum melanjutkan, janganlah dilupakan bahwa kita tetap mengacu kepada sejumlah asumsi yang telah disepakati sebelumnya.

Prefix instalasi

Prefix adalah satu variabel yang lumayan sakti. Pada saat konfigurasi dengan *script configure*, apabila kita menentukan prefix ke suatu direktori, maka, segala file akan ditempatkan di bawah prefix, dengan tetap menyisakan struktur direktori yang digunakan.

Merujuk ke program contoh moc. Berikut ini adalah hasil konfigurasi, yang mengatur prefix ke /tmp/moc.

```
$ ./configure -prefix=/tmp/moc
...
...
...
--------------------------------------
MOC will be compiled with:
Decoder plugins:    mp3 musepack
vorbis flac ffmpeg speex
```

```
OSS:                yes
ALSA:               yes
JACK:               yes
DEBUG:              yes
ICONV:              yes
RCC:                no
Network streams:    yes
Resampling:         no

-----------------------------------
WARNING: since MOC version 2 the
executable file name was changed to
mocp!
Please remove old moc binary if you
have installed an older version.
```

Melanjutkan konfigurasi, kita akan melakukan kompilasi:

```
$ make
...
...
...
mkdir .libs
gcc -g -O2 -Wall -W -I/usr/include/
alsa -o mocp log.o protocol.o
server.o main.o common.o playlist.o
fifo_buf.o out_buf.o audio.o decoder.
o interface.o interface_elements.
o menu.o files.o options.o player.o
playlist_file.o themes.o keys.o ltdl.
o io.o compat.o audio_conversion.
o rbtree.o tags_cache.o utf8.o oss.
o alsa.o jack.o null_out.o io_curl.o
-Wl,--export-dynamic  -pthread /usr/
lib/libasound.so /usr/lib/libjack.so
-L/usr/lib /usr/lib/libcurl.so /usr/
lib/libidn.so -lncursesw -lrt -lm -
lpthread -lssl -lcrypto -ldl -lz
make[2]: Leaving directory `/tmp/
TEMP/moc-2.4.1'
make[1]: Leaving directory `/tmp/
TEMP/moc-2.4.1'
```

Instalasi kemudian, dapat dilakukan dengan memberikan perintah:

```
$ make install
...
...
...
test -z "/tmp/moc/share/man/man1"
|| mkdir -p -- "/tmp/moc/share/man/
man1"
 /usr/bin/ginstall -c -m 644 './
mocp.1' '/tmp/moc/share/man/man1/
mocp.1'
make[2]: Leaving directory `/tmp/
TEMP/moc-2.4.1'
make[1]: Leaving directory `/tmp/
TEMP/moc-2.4.1'
```

Untuk melihat hasil pengaturan prefix, berikut ini adalah isi direktori /tmp/moc (yang dibuat secara otomatis):

```
$ ls -al /tmp/moc/
total 4
drwxr-xr-x  5 nop  users   38 2007-
06-13 21:07 ./
drwxrwxrwt 13 root root  4096 2007-
06-13 21:07 ../
drwxr-xr-x  2 nop  users   17 2007-
06-13 21:07 bin/
drwxr-xr-x  3 nop  users   16 2007-
06-13 21:07 lib/
drwxr-xr-x  5 nop  users   36 2007-
06-13 21:07 share/
```

Bisa kita lihat, bahwa binary moc akan disimpan pada /tmp/moc/bin. Pustaka yang dibutuhkan akan disimpan di /tmp/moc/lib. Share data akan disimpan di /tmp/moc/share. Semuanya di bawah /tmp/moc. Itulah gunanya variabel prefix yang kita atur sebelumnya.

Kita pun dapat menjalankan mocp (executable moc), dengan menjalankan /tmp/moc/bin/mocp. Secara otomatis, mocp akan mencari semua kebutuhannya sesuai prefix yang telah ditentukan. Sebagai contoh, untuk mencari pustaka, mocp akan mencari ke /tmp/moc/lib/.

Uninstall moc juga dapat dilakukan dengan sangat mudah, yaitu dengan menghapus direktori /tmp/moc.

Permasalahan:

- Pertama-tama adalah *path*, khususnya untuk binary *executable* dan manual. Kalau hanya satu program, dengan mudah kita masukkan /tmp/moc/bin ke variabel PATH. Tapi, kalau ada puluhan? Beberapa package management yang menerapkan cara seperti ini menggunakan sistem *symlink*. Apabila Anda tidak keberatan dengan banyaknya symlink di /usr/bin atau /usr/local/bin ke prefix/bin berbagai program, Anda selalu bisa melakukannya dengan bantuan *shell script*.
- Yang kedua, kemungkinan terjadinya redundansi program dan pustaka.
- Permasalahan-permasalahan lain, yang disebabkan berbedanya sistem penanganan program.

Tulisan ini tidak mengajak Anda untuk menentukan prefix yang berbeda untuk

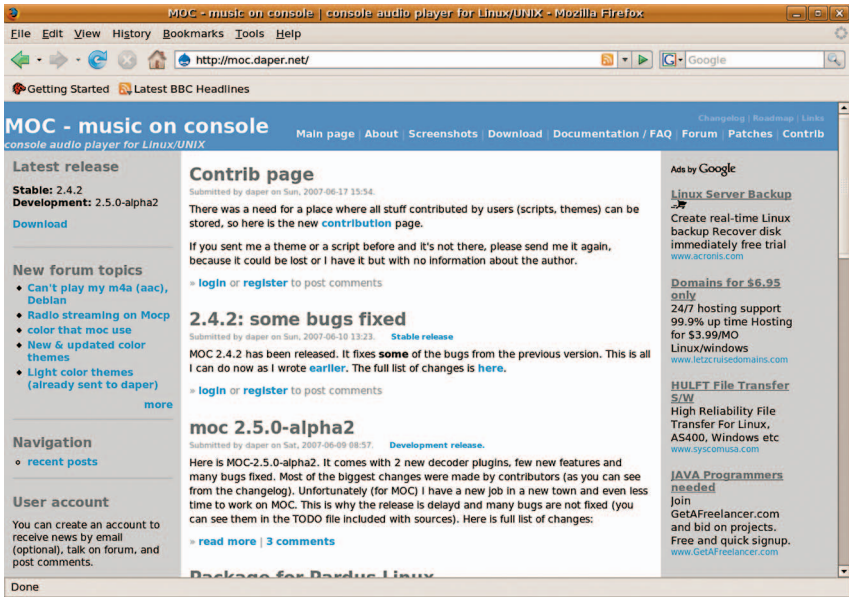
setiap program yang Anda kompilasi sendiri. Bahwa Anda menerima cara seperti ini, hanyalah salah satu cara penanganan software di Linux. Lihatlah pada pembahasan berikut ini.

Prefix yang lebih detail

Script configure juga menawarkan kepada kita untuk mengatur prefix secara lebih detail. Artinya, kita tidak mengatur prefix secara keseluruhan, seperti pada contoh /tmp/moc sebelumnya. Kita bisa mengatur agar file binary ditempatkan di /usr/bin. Kemudian, file pustaka di /usr/lib. Lalu, *share* data di /usr/share/. File konfigurasi di /usr/etc. Apapun yang Anda pilih. Dengan demikian, masih terdapat kemungkinan untuk mempertahankan lokasi binary executable yang Anda inginkan, dan tidak terlalu bermasalah dengan integrasi sistem.

Masih dengan contoh program moc, perhatikanlah beberapa variabel configure berikut ini:

```
$ ./configure --help | grep -i prefix
  --prefix=PREFIX          install
architecture-independent files in
PREFIX
  --exec-prefix=EPREFIX    install
architecture-dependent files in
EPREFIX
                           [PREFIX]
an installation prefix other than
`/usr/local' using `--prefix',
for instance `--prefix=$HOME'.
  --bindir=DIR             user
executables [EPREFIX/bin]
  --sbindir=DIR            system
admin executables [EPREFIX/sbin]
  --libexecdir=DIR         program
executables [EPREFIX/libexec]
  --sysconfdir=DIR         read-only
single-machine data [PREFIX/etc]
  --sharedstatedir=DIR     modifiable
architecture-independent data
[PREFIX/com]
  --localstatedir=DIR      modifiable
single-machine data [PREFIX/var]
  --libdir=DIR             object code
libraries [EPREFIX/lib]
  --includedir=DIR         C header
files [PREFIX/include]
  --datarootdir=DIR        read-only
arch.-independent data root [PREFIX/
share]
  --program-prefix=PREFIX
prepend PREFIX to installed program
names
   --with-libFLAC=PFX    Prefix where
libFLAC is installed (optional)
```



Situs web moc.

Sebagai catatan, keluaran dari script configure tersebut tidaklah sama pada semua program.

Dari berbagai variabel tersebut, kita bisa mengatur lebih detil lokasi instalasi program. Contoh berikut ini akan mengatur agar:

- bindir moc diatur ke /tmp/newroot/usr/bin.
- libdir moc diatur ke /tmp/newroot/usr/lib.
- datarootdir moc diatur ke /tmp/newroot/usr/share.

Berikut ini adalah perintah yang digunakan:

```
$ ./configure --bindir=/tmp/newroot/
usr/bin --libdir=/tmp/newroot/usr/
lib --datarootdir=/tmp/newroot/usr/
share
```

Lakukanlah kompilasi dengan make. dan instalasi dengan make install, maka Anda pun akan mendapatkan file-file moc terinstal di prefix yang Anda tentukan sebelumnya.

Variabel DESTDIR

Pada contoh-contoh sebelumnya, prefix selalu kita berikan pada direktori non standar, di mana dapat pula ditulisi oleh user non-root. Andaikata kita memilih untuk memberikan prefix /usr, yang umumnya digunakan oleh berbagai distro, maka instalasi (make install) haruslah dilakukan oleh user root.

Perlu kita ingat bahwa tidak semua makefile datang dengan rule untuk uninstall. Ketika kita sembarangan melakukan instalasi, sistem akan berpeluang untuk kotor.

Bagaimana caranya agar kita bisa menginstal ke direktori sementara, terlepas dari prefix yang kita atur sebelumnya? Gunakanlah variabel DESTDIR. Kita bisa melakukan konfigurasi dan kompilasi seperti biasa. Hanya, pada saat instalasi yang berbeda:

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

$ make

$ make DESTDIR=/tmp/TEMP/moc install
```

Sebelum instalasi sesuai prefix dilakukan, moc, pada contoh tersebut, diinstall pada /tmp/TEMP/moc. Kita bisa menganalisis sebelum melakukan make install pada prefix yang sesungguhnya.

Linux adalah sistem yang sangat terbuka. Anda selalu bisa memilih cara yang paling cocok dengan kebutuhan Anda. Pembahasan di tulisan ini hanya membahas sedikit, dari berbagai isu seputar instalasi program di Linux. Sampai di sini dulu pembahasan ini. Selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

Traffic Web Server dan Klien pada NS2

NS2 sebagai salah satu simulator jaringan, menyediakan aplikasi untuk melihat proses aliran data dari *web server* ke klien. Kita dapat melihat visualisasi interaksi *cache* server dan klien. Pada artikel ini, akan dibahas cara mengamati traffic web server dan klien pada aplikasi NS2.

Pada tutorial kali ini, tidak lagi dijelaskan hal-hal dasar seperti pada tutorial NS2 yang pertama (*InfoLINUX* 05/2007). Ada tiga kelas utama dalam aplikasi web cache server, klien dan cache. Web klien dapat kita asumsikan sebagai *browser* klien. Satu Klien hanya boleh mempunyai satu cache, sedangkan satu cache dapat menangani multiserver. Sebelum membuat server, cache dan klien kita perlu mendefinisikan *node* untuk setiap server, cache dan klien yang akan dibuat.

```
set server [$ns node]
set client [$ns node]
set cache [$ns node]
set http_server [new Http/Server
$ns
$server]
set http_client [new Http/Client
$ns
$client]
set http_cache [new Http/Cache $ns
$cache]
```

Selain terkoneksi secara fisik (tutorial NS2 di edisi 05/2007), klien, server dan cache harus terkoneksi secara aplikasi. Mulai dan putusnya koneksi dapat dilakukan penjadwalan.

```
$http_client connect $http_cache
$http_cache connect $http_server
$http_client disconnect $http_cache
$http_cache disconnect $http_server
```

Setiap melakukan permintaan halaman, klien dan server menggunakan PagePool untuk mendapatkan nilai ID Halaman. Selain itu, klien juga perlu sebuah variabel acak untuk menghasilkan waktu selang antar-permintaan.

```
$http_client set-page-generator
$pagepool
$http_server set-page-generator
$pagepool
$http_client set-interval-generator
$waktuacak
```

Klien meminta halaman dapat menggunakan dua jenis perintah: *start-session* dan *start*. Start-session meminta kiriman halaman secara acak, sedangkan start melakukan penumpukan di PagePool klien.

```
$http_client start-session $http_
cache $http_server
$http_client start $http_cache
$http_server
```

Objek yang tidak kalah pentingnya pada simulasi kali adalah Pagepool. Pagepool digunakan untuk menghasilkan informasi halaman seperti nama, ukuran, dan waktu terakhir perubahan. Ada lima macam pagepool, yaitu:

1. PagePool/Math.
   PagePool ini adalah yang paling sederhana, di mana hanya bisa menghasilkan satu halaman teks, namun ukuran file halamannya dapat dibuat acak.
   - ranvar-age: nilai acak untuk menghasilkan berapa lama halaman ada.
   - anvar-size: nilai acak untuk menghasilkan berapa ukuran file halaman.
2. PagePool/CompMath.
   PagePool ini kompleks daripada sebelumnya, di mana dapat terdiri atas dua objek dalam setiap halamannya, yaitu objek utama yang berupa teks, dan objek tambahan seperti gambar, dan sebagainya.
   - ranvar-main-age: nilai acak untuk menghasilkan berapa lama halaman teks ada.
   - ranvar-obj-age: nilai acak untuk menghasilkan berapa lama objek ada.
3. PagePool/ProxyTrace.
   PagePool/ProxyTrace membutuhkan dua file sebagai parameternya, yaitu pglog dan reqlog.
   Format pglog:

Format	ID_server	ID_Url	Ukuran_Hal	Jumlah_akses
Contoh	0	1	2788	4

   Format reqlog:

Format	waktu	ID_klien	ID_server	ID_Url
Contoh	0.000	1	0	1

   Pada akhir file reqlog harus ada baris:

Format	i	durasi_waktu	Jumlah_Url
Contoh	i	10	5

   Berikut fungsi-fungsi pada PagePool/ProxyTrace:
   - set-client-num: jumlah klien yang melakukan permintaan.
   - ranvar-dp: nilai acak untuk halaman dinamis.
   - ranvar-sp: nilai acak untuk halaman statis.
   - set-regfile: nama file yang berisi permintaan halaman.
   - set-pgfile: nama file yang berisi informasi halaman.
4. PagePool/Client.

PagePool/Client membantu cache menyimpan jejak halaman yang datang.

5. PagePool/WebTraf
   PagePool ini merupakan modul yang berdiri sendiri, namun masih menggunakan *framework* PagePool. WebTraf digunakan untuk melihat pola traffic web. Berikut perintah Tcl yang penting pada WebTraf:
   - set-num-session.
   - set-num-server.
   - set-num-client.

Selain PagePool, pembuatan variabel acak juga dibutuhkan untuk menghasilkan halaman pada kurung interval tertentu. Variabel acak pada simulasi ini terdapat dua jenis, yaitu Constant dan Exponential. Constant digunakan untuk menghasilkan ukuran halaman, sedangkan Exponential digunakan untuk menentukan lama halaman tersebut ada. Parameter val_ perlu diisi dalam Constant dan avg_ di:

```
set var_con [new RandomVariable/
Constant]
$var_con set val_ 1024
set var_exp [new RandomVariable/
Exponential]
$var_exp set avg_ 5
```

Variabel acak dan PagePool mempunyai hubungan penting dalam simulasi ini, yaitu pada saat server menghasilkan halaman, dan klien meminta halaman. Lebih jelasnya, akan kita lihat pada skrip TCL yang akan kita buat.

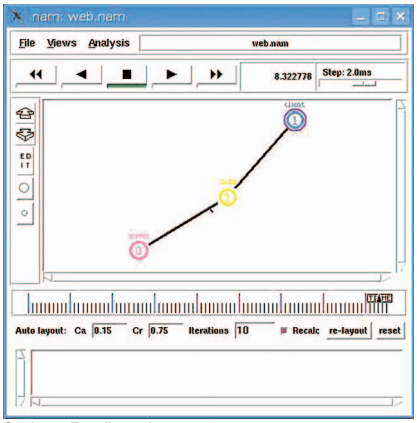
Xgraph merupakan *software* terpisah dari NS2, meskipun terdapat dalam 1 paket NS2, jika kita menginstal NS2 allione. Software X-Windows ini digunakan untuk melihat statistik aliran data pada simulasi NS2. Input Xgraph didapat dari file *trace* (jejak) selama proses simulasi berlangsung. Untuk membuat file jejak tersebut, kita perlu menangkap aliran data dengan menggunakan *trace-queue*.

```
$ns trace-queue $client $cache
$nama_file_jejak
```

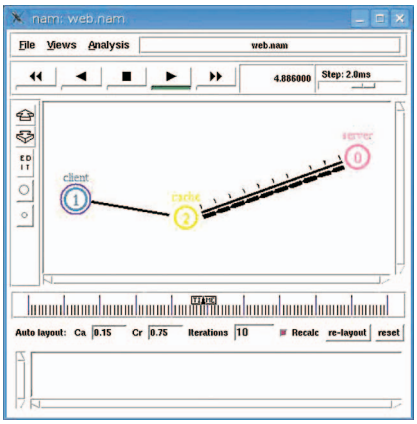
Setelah itu, file tersebut perlu dimodifikasi agar sesuai dengan *input* Xgraph. Pada tutorial ini, *parsing* file dilakukan dengan skrip AWK. Untuk server kita pilih kata "ack", sedangkan untuk klien kita pilih kata "tcp". Nilai sesudah ack.tcp kita jumlahkan, sehingga kita dapatkan jumlah aliran data.

Contoh skrip awk di server:

```
if ($1=="-" && $5=="ack")  {
datasc=datasc + $6
print $2, datasc*8.0/$2/1000000
}
```



Gambar 1. Tampilan web nam pertama.



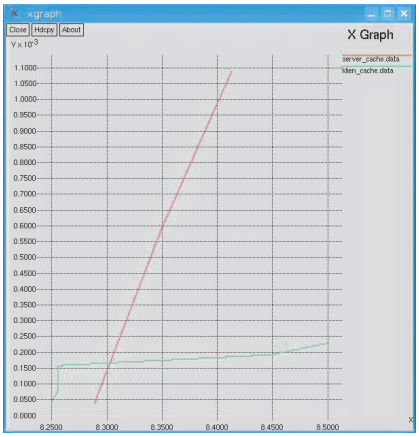
Gambar 2. Tampilan web nam kedua.

Untuk menampilkan hasil pada Xgraph, gunakan perintah:

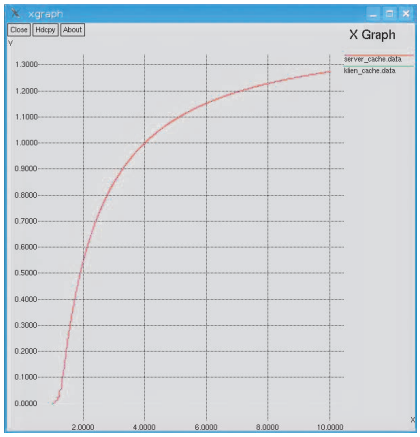
```
#xgraph inputfile1 inputfile2 dst ....
```

Berikut contoh skrip TCL dengan PagePool/Math:

```
#namafile web.tcl
set ns [new Simulator]
set nf [open web.nam w]
$ns namtrace-all $nf
#variabel penghitung aliran data
set datasc 0
set datacc 0
#membuat file jejak
set klien_cache [open klien_cache.
out w]
set server_cache [open server_
cache.out w]
#membuat prosedur hentikan kirim
proc stop_kirim {} {
    global ns nf datasc datacc
    $ns flush-trace
    close $nf
 exec awk {
{
if ($1=="-" && $5=="ack")  {
datasc=datasc + $6
print $2, datasc*8.0/$2/1000000
}
}
} server_cache.out > server_cache.
data
exec awk {
{
if ($1=="-" && $5=="tcp")  {
datacc=datacc + $6
print $2, datacc*8.0/$2/1000000
}
}
} klien_cache.out > klien_cache.
data
exec nam web.nam &
exit 0}
#pembuatan node
set server [$ns node]
set client [$ns node]
set cache [$ns node]
$server label "server"
$client label "client"
$cache label "cache"
#pembuatan hubungan antar node
$ns duplex-link $server $cache
1.5Mb 30ms DropTail
$ns duplex-link $cache $client 10Mb
3ms DropTail
#pembuatan aplikasi
set http_server [new Http/Server
$ns $server]
set http_client [new Http/Client
$ns $client]
set http_cache [new Http/Cache $ns
$cache]
#membuat unicast routing
$ns rtproto Session
#membuat pagepool dan var acak
set pgpm [new PagePool/Math]
#var konstan untuk ukuran hal
set v_kons [new RandomVariable/
Constant]
$v_kons set val_ 1024
$pgpm ranvar-size $v_kons
#var exponential untuk lama halaman
set v_exp [new RandomVariable/
Exponential]
$v_exp set avg_ 5
$pgpm ranvar-age $v_exp
#memberikan nilai page pool ke
server
$http_server set-page-generator
$pgpm
#var exponential untuk interval
```



Gambar 3. Tampilan xgraph math.



Gambar 4. Tampilan xgraph proxy.

```
permintaan klien
set v_exp [new RandomVariable/
Exponential]
$v_exp set avg_ 5
#memberikan nilai page pool ke
  klien
$http_client set-interval-generator
$v_exp
$http_client set-page-generator
$pgpm
#membuat logger
set log [open http.log w]
$http_server log $log
$http_client log $log
$http_cache log $log
#penjadwalan
$ns at 0.0 "$ns trace-queue $client
$cache $klien_cache"
$ns at 0.0 "$ns trace-queue $server
$cache $server_cache"
$ns at 1.0 "mulai"
$ns at 10.0 "stop_kirim"
$ns at 21 "selesai"

proc mulai {} {
                global ns http_
server http_cache http_client
                $http_client
connect
$http_cache
                $http_cache connect
$http_server
                $http_client start-
session $http_cache $http_server}
proc selesai {} {
                global ns log
                $ns flush-trace
                flush $log
                close $log}
$ns run
```

Untuk skrip kedua akan digunakan pagepool jenis Proxy Trace. Kita dapat menggunakan skrip TCL PagePool Math sebelumnya, dan bagian pembuatan PagePool Math sampai dengan sebelum pembuatan *logger* kita ubah menjadi:

```
set pproxy [new PagePool/
ProxyTrace]
#kedua file hrs dibuat sblm simulasi
$pproxy set-pagefile "pglog"
$pproxy set-reqfile "reqlog"
$pproxy set-client-num 1
$pproxy bimodal-ratio 0.1
#var konstan untuk halaman statis
set v_kons [new RandomVariable/
Constant]
$v_kons set val_ 1024
$pproxy ranvar-sp $v_kons
#var exponential untuk halaman
dinamis
set v_exp [new RandomVariable/
Exponential]
$v_exp set avg_ 5
$pproxy ranvar-dp $v_exp
#memberikan nilai page pool ke
server
$http_server set-page-generator
$pproxy
#pada proxytrace tidak prlu
memberikan jangka permintaan
#memberikan nilai page pool ke
klien
$http_client set-page-generator
$pproxy
```

Gambar 1 dan 2 adalah tampilan pada NAM, ketika skrip TCL tersebut dijalankan.

Ketika simulasi telah dilakukan, kita dapat melihat hasil simulasi pada xgraph dengan perintah di bawah ini, dan tampilannya seperti Gambar 3 dan 4.

```
#exec xgraph server_cache.data
klien_cache.data
```

Nur Aini R, Sistem Informasi, FTIF, ITS [iin@its-sby.edu]

Membuat Program ps Sendiri

Program ps sangat berguna bagi kita, untuk mengetahui proses-proses yang ada di sistem. Dengan ps, kita bisa mengetahui banyak informasi tentang proses. Sebagai contoh adalah PID. Dengan mengetahui PID proses, kita bisa melakukan terminasi proses, apabila proses tidak bekerja dengan benar. Di tulisan ini, kita akan membangun sendiri program ps dengan bantuan *shell script*.

Di Linux, terutama bagi yang bekerja di *console*, bukanlah hal yang asing untuk menggunakan program ps dan kill. Dengan program ps, kita bisa mengetahui daftar proses yang ada di sistem. Dengan program kill, kita bisa melakukan terminasi proses.

Program ps sendiri, sebenarnya memiliki sangat banyak kemampuan. Kita bisa mengamati proses-proses di sistem dan dengan sedikit bantuan *tool* sistem, kita bisa menampilkan informasi kalau-kalau terdapat proses yang "mencurigakan". Dengan demikian, kita bisa mengantisipasi program jahat yang berjalan di sistem kita.

Di tulisan ini, kita hanya akan membahas pembuatan program ps yang sangat sederhana, yang memiliki fitur-fitur berikut:

- Dapat menampilkan daftar proses terurut PID.
- Dapat menampilkan status proses.
- Dapat secara detail menampilkan program yang dijalankan, lengkap dengan argumen yang diberikan.

Lebih kurang, apabila mempergunakan program ps, berikut ini adalah *output* yang ingin dicapai oleh program kita:

```
$ ps axo pid,stat,args
  PID STAT COMMAND
    1 Ss   init [3]
    2 SN   [ksoftirqd/0]
    3 S<   [events/0]
    4 S<   [khelper]
    5 S<   [kthread]
    8 S<   [kblockd/0]
    9 S<   [kacpid]
   92 S<   [khubd]
   94 S<   [kseriod]
...
...
  864 S<   [xfsbufd]
  866 S<   [xfssyncd]
 1030 S<s  /sbin/udevd --daemon
 2052 S<   [pccardd]
 2053 S<   [pccardd]
 2132 S<   [reiserfs/0]
 2186 Ss   /usr/sbin/syslogd
 2189 Ss   /usr/sbin/klogd -c 3 -x
 2331 Ss   /usr/sbin/inetd
...
...
...
```

Hanya, untuk status proses, kita akan menampilkan deskripsi proses (contoh: *running, sleeping*), dan bukan dengan simbol seperti S,Ss, dan lainnya.

/proc

Untuk membangun program ps, kita sangat tergantung dengan /proc file sistem. Di /proc, kita akan mendapatkan banyak direktori yang memiliki nama berupa angka-angka. Sebagai contoh:

```
$ ls -1 /proc/ | grep -i ^[0-9]|
head -n5
1/
1030/
141/
142/
143/
```

Setiap direktori mewakili satu proses, di mana nama direktori adalah PID. Direktori-direktori PID di /proc bersifat dinamis, di mana akan menghilang begitu proses selesai, dan akan ditambahkan begitu terdapat proses baru.

Sebagai contoh, kita akan masuk ke direktori 1, yang merupakan PID dari init. Berikut ini adalah isi direktori /proc/1 di sistem penulis:

```
# ls -al  /proc/1/
total 0
dr-xr-xr-x  5 root root 0 2007-06-15
14:35 .
dr-xr-xr-x 63 root root 0 2007-06-15
14:35 ..
dr-xr-xr-x  2 root root 0 2007-06-15
08:36 attr
-r--------  1 root root 0 2007-06-15
08:34 auxv
-r--r--r--  1 root root 0 2007-06-15
07:55 cmdline
lrwxrwxrwx  1 root root 0 2007-06-15
08:34 cwd -> /
-r--------  1 root root 0 2007-06-15
08:34 environ
lrwxrwxrwx  1 root root 0 2007-06-15
08:34 exe -> /sbin/init
dr-x------  2 root root 0 2007-06-15
08:36 fd
-r--r--r--  1 root root 0 2007-06-15
08:34 maps
```

```
-rw-------  1 root root 0 2007-06-15
08:34 mem
...
...
lrwxrwxrwx  1 root root 0 2007-06-15
08:34 root -> /
-rw-------  1 root root 0 2007-06-15
08:34 seccomp
-r--r--r--  1 root root 0 2007-06-15
08:34 smaps
-r--r--r--  1 root root 0 2007-06-15
14:35 stat
-r--r--r--  1 root root 0 2007-06-15
08:34 statm
-r--r--r--  1 root root 0 2007-06-15
14:35 status
dr-xr-xr-x  3 root root 0 2007-06-15
08:36 task
-r--r--r--  1 root root 0 2007-06-15
08:34 wchan
```

Terdapat sejumlah file/direktori, yang di antaranya berfungsi untuk memberikan informasi proses kepada kita. Kita akan memperhatikan dua file berikut:

- cmdline: berisikan *command line* proses.
- status: status umum proses.

Contoh:

```
# cat /proc/1/cmdline
init [3]

# cat /proc/1/status
Name:   init
State:  S (sleeping)
SleepAVG:       98%
Tgid:   1
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 32
Groups:
...
...
```

Dari file status, kita akan membutuhkan informasi:

- *Name*.
- *State*.

Dengan modal tersebut, kita akan membangun program ps sendiri.

NPS: program ps kita sendiri

Berikut ini adalah *source code* shell script nps. Pembahasan akan kita lakukan setelah source code:

```
#!/bin/sh

PIDS=`ls -1 /proc | grep -i ^[0-9]
| sort -n`
for pid in $PIDS
do
  if [ -d /proc/$pid ]
  then
    cmdline="`cat -v /proc/$pid/
cmdline | tr '^@' ' ' 2>/dev/null`"
    if [ ! -z "$cmdline" ]
    then
      cmdstr=$cmdline
    else
      pname="`cat /proc/$pid/status
| grep -i Name: | cut -d: -f2 | cut
-d' ' -f2 | tr -d '[:space:]'`"
      cmdstr="[$pname]"
    fi

    state="`cat /proc/$pid/status |
grep -i State: | cut -d: -f2 | cut
-d' ' -f2`"
    printf "%6d %-12s %s\n" "$pid"
"$state" "$cmdstr"
  fi
done
```

Berikanlah hak akses *executable*, dengan perintah berikut:

```
$ chmod +x nps
```

Berikut ini adalah contoh output program:

```
$ ./nps
     1 (sleeping)   init [3]
     2 (sleeping)   [ksoftirqd/0]
     3 (sleeping)   [events/0]
     4 (sleeping)   [khelper]
     5 (sleeping)   [kthread]
...
...
...
   864 (sleeping)   [xfsbufd]
   866 (sleeping)   [xfssyncd]
  1030 (sleeping)   /sbin/udevd -
-daemon
  2052 (sleeping)   [pccardd]
  2053 (sleeping)   [pccardd]
  2132 (sleeping)   [reiserfs/0]
  2186 (sleeping)   /usr/sbin/
syslogd
  2189 (sleeping)   /usr/sbin/klogd
-c 3 -x
  2331 (sleeping)   /usr/sbin/inetd
...
...
...
```

Penjelasan source code:

- Pertama-tama, kita akan dapatkan terlebih dahulu semua direktori angka (tepatnya: nama direktori diawali angka) di /proc, yang kemudian kita *sort* secara numerik:

```
PIDS=`ls -1 /proc | grep -i ^[0-
9] | sort -n`
```

- Untuk setiap direktori PID yang kita dapatkan, apabila direktori tersebut ditemukan (kita harus memeriksa lagi, karena proses sangat dinamis):
  - Kita akan membaca file /proc/<PID>/cmdline (dan membuang karakter yang tidak diperlukan).

```
cmdline="`cat -v /proc/$pid/
cmdline | tr '^@' ' ' 2>/dev/
null`"
```

  - Apabila file cmdline tersebut tidak kosong, maka kita akan menampilkan informasi yang terdapat di file tersebut.

```
cmdstr=$cmdline
```

  - Apabila file cmdline tersebut kosong, maka kita akan membaca Name dari /proc/<PID>/status, dan menampilkannya dalam format [<NAME>], mirip dengan program ps.

```
pname="`cat /proc/$pid/
status | grep -i Name: | cut -
d: -f2 | cut -d' ' -f2 | tr -d
'[:space:]'`"
cmdstr="[$pname]"
```

  - Kita kemudian membaca informasi State dari /proc/<PID>/status, dan mengambil deskripsi status (sleeping, running dan lainnya):

```
state="`cat /proc/$pid/
status | grep -i State: | cut
-d: -f2 | cut -d' ' -f2`"
```

  - Terakhir, kita menampilkan semua informasi yang telah didapatkan menggunakan program printf. Dengan menggunakan printf, kita bisa memformat teks yang ingin ditampilkan dengan rapi.

```
printf "%6d %-12s %s\n"
"$pid" "$state" "$cmdstr"
```

Demikianlah. Kita sudah membangun program yang mirip dengan ps, walaupun program yang kita buat ini berjalan jauh lebih lambat. Silakan dikembangkan lagi sesuai kebutuhan. Selamat mencoba!

Noprianto [noprianto@infolinux.co.id]

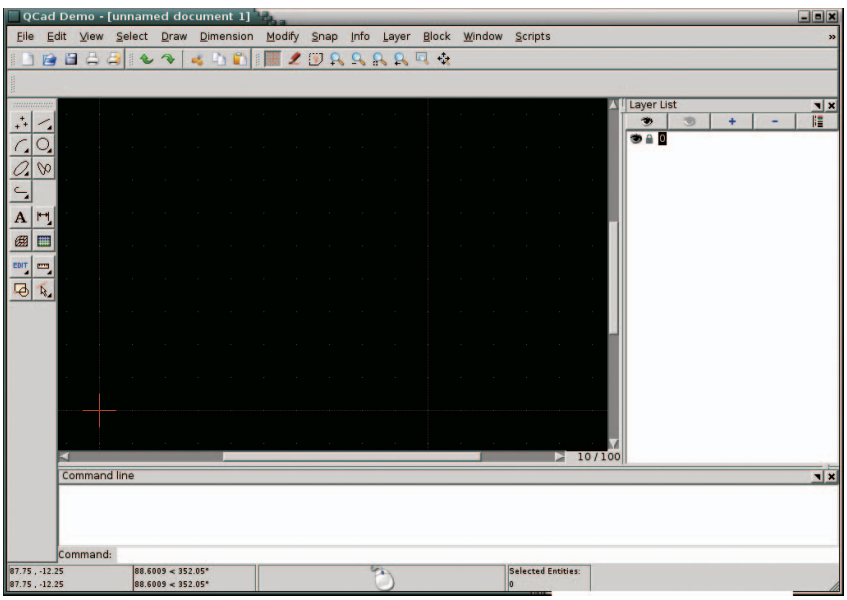
QCad Aplikasi Rancang Bangun

QCad merupakan aplikasi program CAD yang mampu berjalan pada dua sistem operasi, yaitu Linux dan Windows. Aplikasi ini dapat berfungsi seperti AutoCAD yang terdapat di platform Windows. Dengan demikian, tidak perlu berkecil hati bagi para pengguna OS Linux yang ingin membuat proyek-proyek yang berhubungan dengan rancang bangun.

Bagi para *drafter*, aplikasi rancang bangun seperti AutoCad bukan merupakan hal yang asing lagi. Peranan AutoCad sangat membantu dalam hal menyelesaikan rancangan proyek-proyek yang berkaitan dengan rancang bangun (*engineering*). Namun karena terbentur oleh permasalahan lisensi yang begitu mahal, akhirnya masih banyak perusahaan-perusahaan yang kesulitan dalam permasalahan lisensi tersebut. Lantas, apakah ada pengganti AutoCad yang walaupun tidak bersifat *free*, namun tidak begitu mahal? Ada, jawabannya QCad.

Pada QCad versi sebelumnya, QCad merupakan aplikasi rancang bangun yang dikhususkan bagi pengguna Linux semata, dan masih bersifat free. Namun sejak QCad 2.1, QCad tidak lagi hanya dikhususkan bagi pengguna Linux saja, melainkan dapat juga digunakan pada sistem operasi yang lain seperti Windows, dan tidak bersifat free. Gambar 1 merupakan tampilan utama dari aplikasi QCad.

Terdapat komponen-komponen pokok yang sangat berperan dalam proses pembuatan objek-objek rancang bangun 2D pada QCad, yaitu di antaranya: *toolbox, command line*, area gambar.



Gambar 1. Tampilan utama QCad.

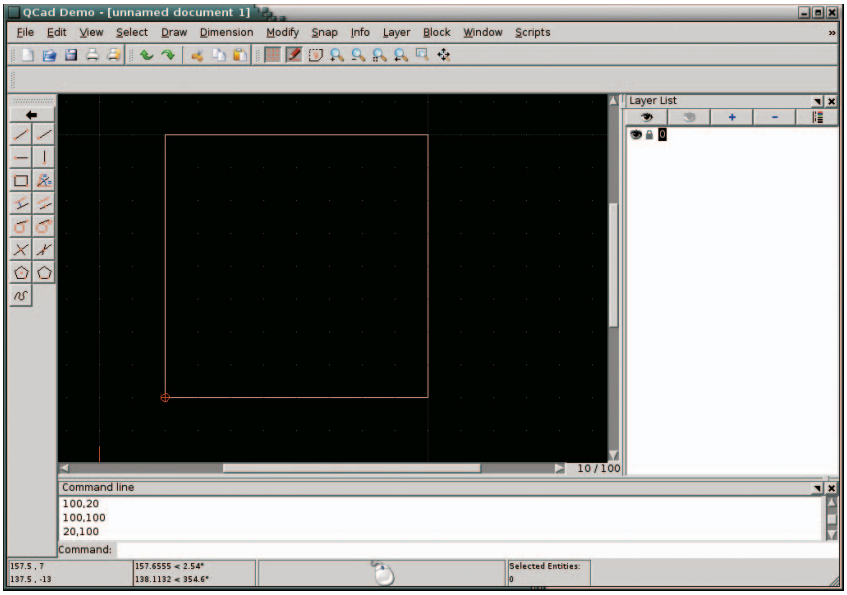
Toolbox

Gambar 2. Toolbox.

Berisikan tool-tool, yang masing-masing tool-tool tersebut mewakili tool-tool di dalamnya, misalnya tool *Line* mewakili *tool Line with two point*, *Horizontal Line*, *Vertical Line*, dan tool-tool lainnya. Gambar 2 merupakan tampilan dari toolbox.

- *Draw points*, berfungsi untuk menentukan dan menggambar point.
- Line, berfungsi untuk membuat garis.
- *Arcs*, berfungsi untuk membuat arc atau sebagian lingkaran.
- *Circle*, berfungsi untuk membuat lingkaran.
- *Ellipse*, berfungsi untuk membuat objek ellip atau lonjong.
- *Splines*, berfungsi untuk membuat suatu kurva terbuka atau tertutup.
- *Polylines*, memiliki fungsi yang sama dengan lines, yaitu untuk membuat garis, namun perbedaannya yaitu garis yang terbentuk oleh polylines menjadi satu kesatuan utuh. Sedangkan garis yang terbentuk oleh line tidak bersifat satu kesatuan utuh.
- *Text*, berfungsi untuk membuat objek teks yang umumnya digunakan untuk memberikan keterangan pada bagian-bagian tertentu.
- *Dimensions*, berfungsi untuk menentukan dimensi pada suatu objek, apakah itu ukuran garis, diameter suatu ling-

IKLAN



Gambar 3. Contoh gambar kotak dengan koordinat kartesius.

karan dan sebagainya.

- *Hatches*, berfungsi untuk mengarsir suatu area pada suatu objek atau area yang merupakan bagian irisan dari kedua buah objek yang saling bersinggungan.
- *Raster images*, berfungsi untuk menyisipkan suatu image yang ber-*extension* tertentu.
- *Edit*, untuk menampilkan tool-tool yang berkaitan dengan pengeditan suatu objek apakah itu tool *Move, Rotate, Scale, Mirror,* dan sebagainya.
- *Measure*, berperan dalam menampilan alat pengukur jarak atau sudut.
- *Create Block*, berperan dalam memblok area tertentu, yang terdiri dari *Deselect all, Select all* dan tool pemblokkan lainnya.
- Select, memiliki peran yang sama dengan tool Create Block, yang membedakan hanya pada tool Select dapat memilih per objek, dari suatu objek yang kompleks.

Command line

Command merupakan tempat di mana kita dapat menyisipkan perintah-perintah yang terdapat pada suatu toolbox. Misalnya kita ingin membuat suatu garis, cukup dengan mengetikan Line pada command atau perintah lainnya. Menurut penulis, command lebih praktis jika dibandingkan dengan menggunakan tool, jika diibaratkan seperti melakukan penginstalan lewat *console* dibanding secara grafis.

Area gambar

Merupakan area atau tempat di mana objek dibuat. Secara default, warna dari area gambar tersebut berwarna hitam, dan terlihat *grid* di sekitarnya. Grid-grid tersebut apabila dianggap mengganggu, maka dapat dihilangkan dengan cara mengklik *View* > *Grid*.

Menggambar objek garis

Terdapat suatu hal yang perlu diperhatikan dalam proses pembuatan suatu objek, terutama objek garis. Hal yang dimaksud adalah koordinat. Seperti halnya AutoCad atau aplikasi program rancang bangun lainnya, QCad pun mengenal tiga sistem pengaturan koordinat, yaitu koordinat kartesius, koordinat relatif, dan koordinat polar.

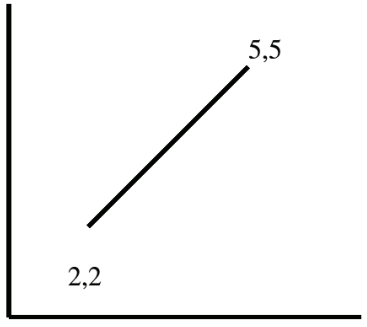
- Koordinat Kartesius.
  Yang dimaksud dengan koordinat relatif, yaitu suatu koordinat yang mengacu pada pertemuan antara titik pada sumbu x dan y. Adapun bentuk umum penulisannya adalah x,y. Misalnya kita akan membuat sebuah kotak dengan menggunakan sistem koordinat kartesius.

```
Command : line
Specify first point : 20,20
Specify next point : 100,20
Specify next point or undo :
100,100
Specify next point or [close/
undo] : 20,100
Specify next point or [close/
undo] : Close
```

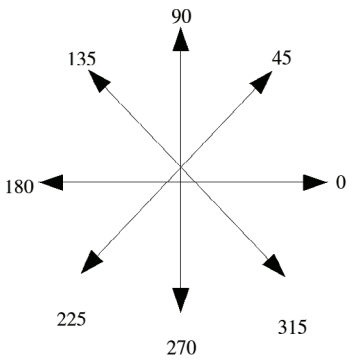
  Kemudian, tekan *ESC* 2x untuk menormalkan kembali. Hasilnya akan terlihat seperti gambar 3.
- Koordinat Relatif.
  Suatu koordinat yang mengacu pada relativitas antara titik pertama terhadap titik berikutnya. Adapun bentuk umum penulisannya adalah @x,y. Untuk lebih memahami tentang sistem koordinat relatif, perhatikan gambar 4.
  Untuk membentuk garis tersebut di atas menggunakan sistem koordinat relatif adalah sebagai berikut ini:

```
Command : line
Specify first point : 2,2
Specify next point : @3,3
Specify next point or undo :
```

  Kemudian tekan ESC 3x untuk menormalkan kembali.
  Pada umumnya, koordinat relatif sering digunakan dalam hal pemetaan, *steplan*, kontur tanah dan proyek-proyek lainnya.
- Koordinat Polar.
  Koordinat polar merupakan koordinat yang sangat sering digunakan oleh para *drafter* dalam membuat sebuah objek. Prinsip kerja dari koordinat polar itu sendiri adalah panjang garis yang disesuaikan dengan sudut. Adapun bentuk penulisannya adalah panjang garis<sudut.



Gambar 4. Sistem koordinat relatif.



Gambar 5. Gambar sudut-sudut rotasi

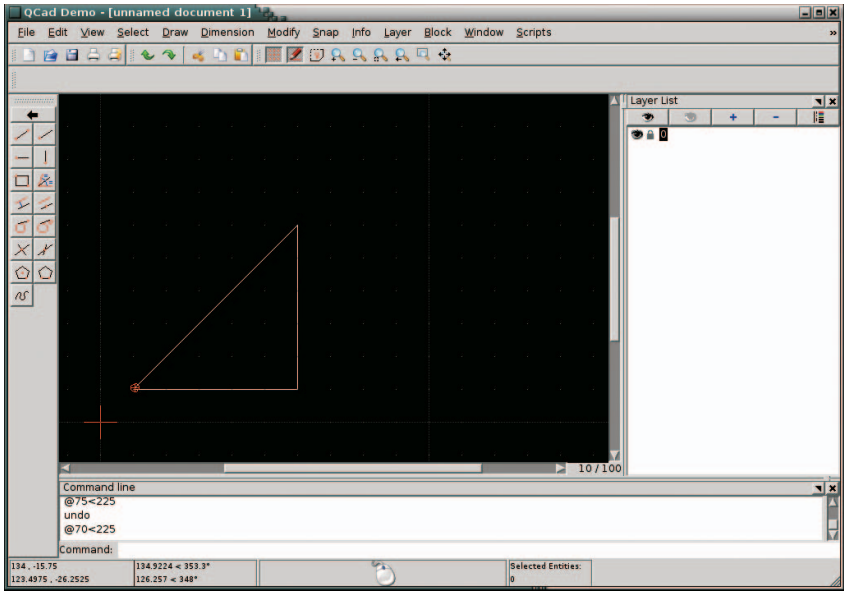
Gambar 5 merupakan gambar dari sudut-sudut rotasi tersebut.

Berikut ini merupakan contoh proses pembuatan objek garis dengan menggunakan koordinat polar. Perintahnya seperti berikut ini:

Command : line

```
Specify first point : 10,10
Specify next point : @50<0
Specify next point or undo :
@50<90
Specify next point or [close/
undo] : @70<225
Specify next point or [close/
undo] : Close
```

Kemudian tekan ESC 3x untuk menormalkan kembali. Hasilnya akan terlihat seperti gambar 6.



Gambar 6. Contoh gambar segitiga dengan koordinat polar.

Menggambar objek rectangle

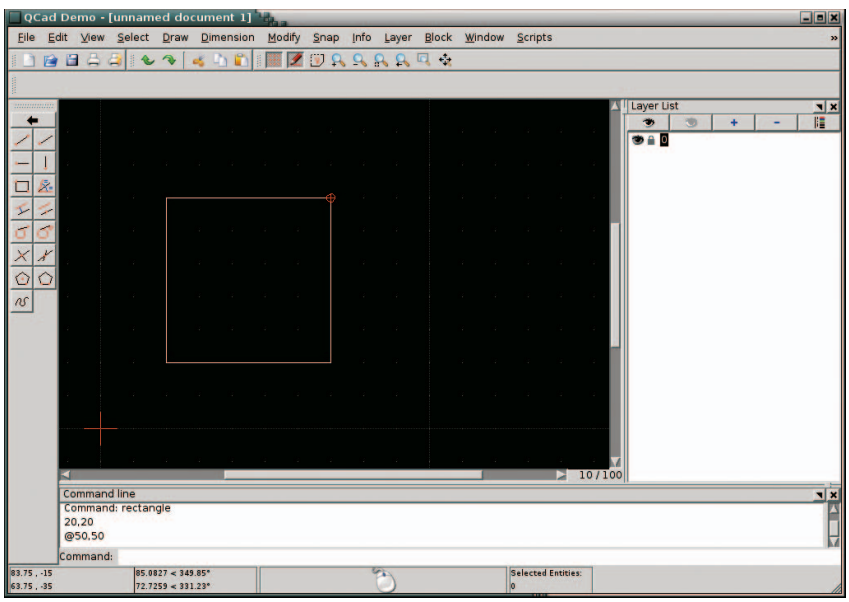
Tool *rectangle* termasuk ke dalam kategori tool Lines, jika Anda ingin membuat rectangle melalui tool, maka cara kerjanya, yaitu:

1. Klik tool Lines, maka terlihat tool-tool lainnya, dan kemudian klik tool Rectangle. Kemudian sisipkan nilai pada *Specify first corner* (dengan menggunakan koordinat kartesius, atau dengan mengklik *mouse* di sembarang posisi).
2. Setelah menyisipkan Specify first corner, maka selanjutnya Anda diminta untuk menyisipkan *Specify second corner* (dengan menggunakan koordinat relatif).

Berikut ini merupakan contoh pembuatan rectangle dengan perintah command line secara langsung:

```
Command : rectangle
Specify first corner : 20,20
Specify second corner : @50,50
```

Kemudian tekan ESC 2x untuk menormalkan kembali. Hasilnya akan terlihat seperti gambar 7.



Gambar 7. Contoh hasil pembuatan objek kotak dengan perintah rectangle.

Menggambar Objek Circle

Circle memiliki rumpun atau kategori menu yang tidak tergolong Line, seperti Rectangle. Berikut ini penjelasan beberapa sub menu circle:

Circle with center and point, yaitu membentuk objek lingkaran, dengan cara menentukan terlebih dahulu titik pusat dari lingkaran tersebut, kemudian menetukan besaran atau radius dari objek lingkaran tersebut.

Circle with center and radius, yaitu membentuk lingkaran dengan hanya menentukan spesifik pusat lingkaran, dengan cara mengetikkan koordinat kartesius dan radiusnya.

Circle with two opposite points, yaitu membentuk lingkaran dengan cara menentukan dua titik. Titik pertama atau first point dapat dilakukan dengan mengklik sembarang area dengan mouse, atau dapat juga dengan menggunakan sistem koordinat kartesius. Setelah itu, penentuan titik kedua juga dapat dilakukan seperti pada penentuan titik pertama.

Circle with three points, yaitu membuat lingkaran dengan menggunakan sistem tiga titik. Cara pembuatannya hampir sama dengan Circle with two opposite point. Yang membedakannya, pada Circle with three points membutuhkan titik ketiga yang dijadikan titik akhir atau titik penentu pembentukan objek lingkaran tersebut. Penetuan titik ke titik tetap dapat dilakukan dengan mengklik pada sembarang area dengan mouse atau dengan menggunakan koordinat kartesius.

Concentric, submenu Concentric memiliki peran hanya sebagai perantara di dalam pembentukan objek-objek lain. Cara kerjanya dengan mengklik submenu Concentric, kemudian masuk ke submenu Line yang terdiri atas line with two points, rectangle, polygon, dan objek lainnya.

Mohamad Sukarno [kamo180372@gmail.com]

Tip dan Trik Zencafe

Zencafe merupakan salah satu distro yang dikhususkan untuk kebutuhan warnet. Desain distro ini sudah cukup sesuai dengan kondisi yang terdapat di warnet. Salah satunya adalah dengan disertakannya *billing* warnet ccl, dan autorecovery tool yang dapat berfungsi seperti aplikasi deepfreze.

Dari beberapa pilihan distro yang cocok digunakan untuk warnet, salah satu yang bagus untuk digunakan adalah Zencafe. Distro buatan lokal dengan kualitas internasional ini memiliki tampilan yang mirip Windows Vista, dan tidak membutuhkan *resource hardware* yang terlalu tinggi. Sehingga tak heran jika distro ini sangat cocok digunakan oleh kalangan warnet yang tidak memiliki spesifikasi hardware yang tinggi.

Pada tutorial kali ini, *InfoLINUX* akan menjelaskan sejumlah tip dan trik yang terdapat pada distro Zencafe, untuk menambah fungsionalitas saat bekerja menggunakan distro ini.

Sharing file dengan Samba

Salah satu kebutuhan warnet adalah dapat saling berbagi folder atau file di dalam jaringan. Untuk menjembatani hal ini, kita dapat menggunakan Samba yang dapat kita gunakan untuk saling berbagi folder dengan sesama pengguna sistem operasi Linux ataupun Windows. Berikut langkah untuk berbagi folder di Zencafe.

1. Pastikan Anda telah mengaktifkan service Samba pada saat kali pertama menginstal. Jika belum mengaktifkan, Anda dapat mengaktifkan *service* Samba dari *start Menu -> System -> Zenpanel*. Setelah tampil halaman Zenpanel, klik menu *Startup Services*.
2. Selanjutnya, kita akan mengonfigurasikan Samba dengan menggunakan SWAT (*Samba Web Administration Tool*). *Login* sebagai *user root*, dan masukkan *password root* Anda di SWAT.
3. Klik *icon* GLOBAL yang terdapat pada SWAT, kemudian isikan beberapa parameter yang terdapat pada menu tersebut, sebagai berikut:
   - *Workgroup*. Isikan dengan nama workgroup yang Anda inginkan.
   - *Netbiosname*. Isikan dengan nama PC Anda.
   - *Security*. Ubah pilihannya menjadi "SHARE".
   - Setelah beberapa poin di atas selesai dilakukan, klik *button "Commit Changes*".
4. Klik icon SHARES, kemudian ubah beberapa parameter berikut:
   - *Enter share name*. masukkan nama share folder yang diinginkan. Misal: *tes_sharing*, kemudian klik *"Create Share"*.
   - *Path*. Pada *option* path, ubah letak path direktori yang ingin Anda share. Misal: */home/supriyanto/Documents*.
   - *Read only*. Ubah pilihan menjadi *No*.
   - *Guest ok*. Ubah pilihan menjadi *Yes*.
   - Terakhir, klik button *"Commit Changes*".
5. Ubah *permission* folder yang telah Anda pilih untuk di-*sharing*, menjadi RWX (777). Untuk melakukan hal ini, jalankan aplikasi Thunar File Manager dari *start Menu -> Accessories -> Thunar File Manager*. Klik kanan folder yang telah Anda pilih untuk di-sharing. Setelah tampil halaman *Document Properties*, Anda ubah pilihan *Access* untuk *Owner, Group*, dan *Other* direktori tersebut menjadi *Read & Write*. Setelah itu, klik *Close*.
6. Sekarang folder yang sudah Anda share, dapat segera diakses oleh komputer lain yang terdapat dalam jaringan.

Setting Samba Client di Zencafe

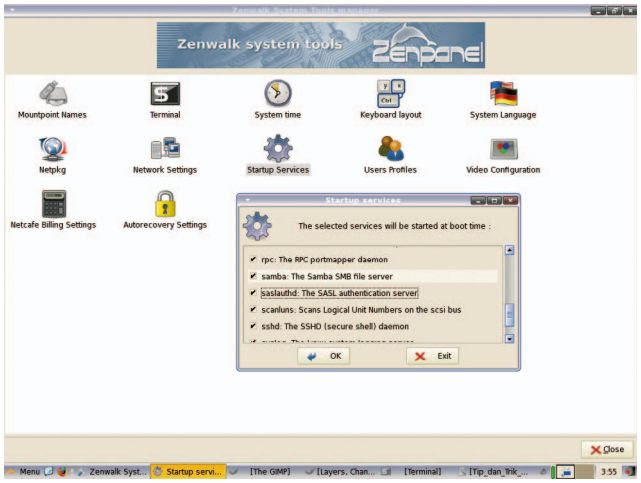
Setelah sebelumnya kita belajar cara menjalankan folder sharing, berikutnya kita akan mencoba untuk mengakses folder sharing dengan menggunakan Samba Client di Zencafe.

Untuk memulai langkah ini, Anda dapat memilih *start Menu -> Network -> FuseSMB-Tools*. Ubah direktori *MountPoint* menjadi direktori *Network* Anda. Dalam contoh ini, dimisalkan direktori MountPoint adalah */home/supriyanto/smb*. Setelah *OK*, klik *Save*. Maka pada kotak *Status*, seharusnya sudah berubah menjadi *SMB is mounted*.

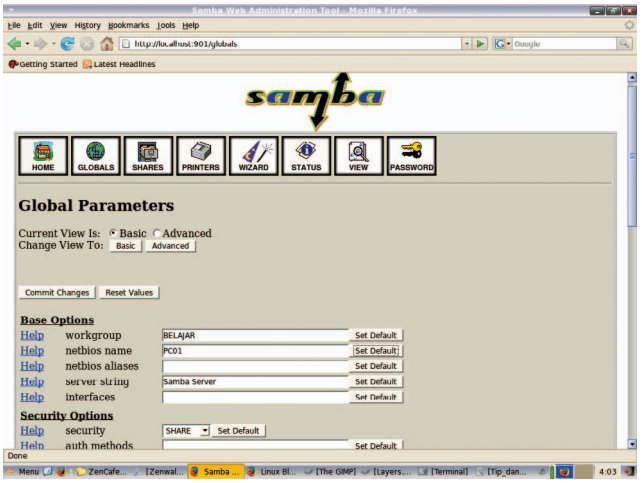
Untuk dapat segera melihat share folder yang terdapat dalam network Anda, klik Network *shortcut* yang terdapat pada *desktop* Anda, setiap kali ingin melihat atau mengakses share folder.

Menggunakan Floppy di Zencafe

Meski penggunaan USB Flash disk sebagai media penyimpanan sudah banyak menggantikan fungsi floppy disk yang rentan rusak, namun terkadang masih ada pelanggan warnet yang ingin mengakses floppy disk. Secara *default*, Zencafe tidak menyediakan



Meng-enable service samba dari Zenpanel.



Beberapa perubahan parameter pada menu Global yang terdapat di SWAT.

fitur akses ke floppy disk setelah proses instalasi selesai. Untuk dapat mengakses floppy disk pada distro Zencafe, Anda dapat melakukan langkah sebagai berikut.

1. Login sebagai root, kemudian buat direktori */mnt/floppy*.

```
$ su -
password: <masukkan password root Anda>

# mkdir -p /mnt/floppy
```

2. Berikutnya, jalankan Thunar File Manager. Dari aplikasi Thunar File Manager, klik menu *Edit -> Configure custom actions*.
3. Pada halaman *Custom Actions*, Klik *(+)* button untuk menambahkan *new custom action*.
4. Setelah tampil halaman *Create Action*, kita akan membuat *item* action di Thunar File Manager, yang nantinya dapat digunakan untuk *mount* floppy disk. Isikan dengan beberapa parameter sebagai berikut pada *Tab Basic*, yang terdapat di halaman Create Action.
   - *Name : Mount Floppy*
   - *Description:*
   - *Command: mount %f*

   Pindah ke Tab *Appearance Conditions* yang masih terdapat pada halaman Create Action. Lakukan hal sebagai berikut:
   - Pada option parameter *File Pattern*, isikan dengan fdfloppy.
   - Pada option *Appears if selection contains*, beri tanda centang hanya pada pilihan *Directories*. Untuk pilihan lainnya, kosongkan saja pilihannya.
5. Kita juga akan membuat sebuah item action di Thunar File Manager yang kita gunakan untuk *unmount* floppy disk. Lakukan langkah nomor 1-4. Hanya saja pada langkah 4, isikan parameter sebagai berikut pada Tab *Basic Action* yang terdapat di halaman Create Action.
   - Name : Unmount Floppy
   - Description:
   - Command: umount %f

   Pindah ke Tab Appearance Conditions yang masih terdapat pada halaman Create Action. Lakukan hal sebagai berikut:
   - Pada option parameter *File Pattern*, isikan dengan fdfloppy.
   - Pada option *Appears if selection contains*, beri tanda centang hanya pada pilihan *Directories*. Untuk pilihan lainnya, kosongkan saja pilihannya.

Sekarang setiap kali Anda ingin mengakses floppy disk, Anda cukup mengklik kanan shortcut floppy yang terdapat pada *Computer directori*. Klik *"Mount Floppy"* untuk melakukan mount floppy disk, dan *"Unmount Floppy"* untuk melakukan unmount floppy disk.

3D Desktop dengan Beryl

Micsosoft Windows Vista telah memiliki 3D desktop dan *themes* yang enak dilihat mata. Namun bukan hanya Windows yang dapat memiliki 3D desktop yang menarik. Desktop Linux juga dapat kita jadikan sebagai 3D desktop. Buat user warnet Anda terpana dengan tampilan 3d desktop yang ada di Linux, dan merasa senang untuk menggunakannya.

Pada bagian ini, kita akan mencoba instalasi Beryl di desktop Zencafe. Untuk percobaan ini, Anda membutuhkan 3D VGA Card, dan paket Beryl. Sebelum dapat menginstalasi Beryl, pastikan Anda telah menginstal *driver* 3D VGA Card pada PC Anda. Jika Anda pengguna VGA Card NVIDIA, instalasikan paket nvidia driver yang dapat Anda peroleh di url *http://www.nvidia.com*. Sedangkan jika Anda pengguna VGA Card ATi, Anda dapat memperoleh driver ATi pada url *http://ati.amd.com*. Silakan merujuk ke halaman Zenwalk Wiki yang berada di *http://wiki.zenwalk.org*, untuk petunjuk detail instalasi driver 3D VGA Card tersebut.

Langkah selanjutnya, login ke XFCE *session* sebagai root, dan jalankan aplikasi *Terminal*. *Backup* file xorg.conf terlebih dahulu, untuk menyimpan konfigurasi xorg, jika sewaktu-waktu terjadi suatu masalah:

```
# cp /etc/X11/xorg.conf /etc/X11/xorg.conf.bak
```

Selanjutnya, edit file xorg.conf tersebut:

```
# vim /etc/X11/xorg.conf
```

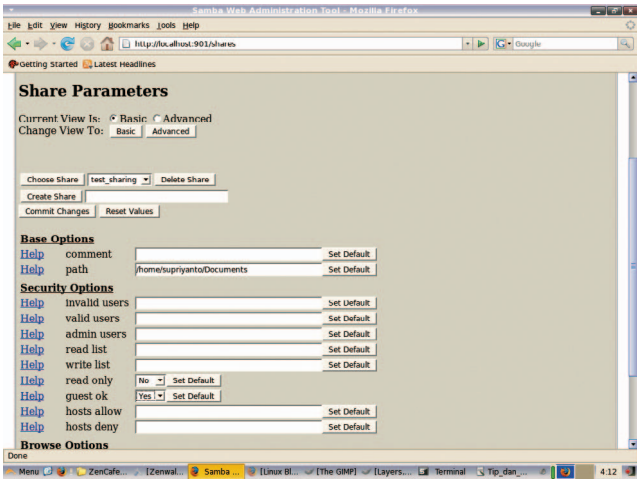
Tambahkan baris berikut pada Section "Device":

```
Option "AddARGBGLXVisuals" "true"
Option "TripleBuffer" "true"
```

Tambahkan juga baris di bawah ini, pada bagian akhir file xorg.conf:

```
Section "Extensions"
    Option "Composite" "Enable"
EndSection
```

Simpan hasil perubahan yang telah dilakukan, dan keluar dari editor vim. Berikutnya, Anda perlu men-*download* paket-paket



Mendefinisikan folder yang akan di-share.

Menginstal autorecovery system di Zencafe.

Beryl untuk distro Slackware, dan semua *plug-ins* yang dibutuhkan dari url *http://web.tiscali.it/meskalamdug/aiglx-en.html*.

Berikut beberapa paket Beryl yang dibutuhkan untuk melakukan uji coba:

1. Beryl.
2. Emerald themes.
3. Emerald.
4. Beryl plugins.
5. Beryl-settings.
6. Beryl-manager.
7. Seom.
8. Beryl-vidcap.
9. Libwnck.
10. Aquamarine.

Buat sebuah direktori, kemudian download semua paket Beryl yang dibutuhkan ke direktori tersebut. Pindah ke direktori tersebut, kemudian instalasikan dengan menggunakan perintah berikut:

```
# installpkg *.tgz
```

Setelah semuanya terinstalasi dengan baik. Sebelum Anda menjalankan Beryl secara langsung, *kill* terlebih dahulu xfwm4, karena Beryl memiliki windows manager sendiri.

```
# killall xfwm4
```

Selanjutnya, jalankan perintah beryl-manager untuk segera menggunakan desktop beryl.

```
# beryl-manager
```

Catatan: Pada saat kali pertama kita menjalankan beryl-manager, akan terdapat sebuah *bug* seputar perpindahan desktop, yang membuat kita kesulitan untuk mengklik icon aplikasi yang terdapat pada desktop. Untuk menyelesaikan hal ini, buka desktop panel, kemudian pilih *Settings -> Desktop Settings*. Lakukan *unselect* dan *reselect 'Allow Xfce to manage the desktop'*.

Mengembalikan taskbar Xfce yang hilang

Suatu saat, entah karena suatu kecerobohan atau karena tak disengaja, *taskbar* yang terdapat di Xfce tak sengaja kita hilangkan. Untuk mengembalikan kembali taksbar Xfce yang hilang, Anda dapat melakukan hal sebagai berikut:

1. Klik kanan desktop, klik menu *Settings -> Settings Manager*.
2. Setelah tampil halaman *Xfce Settings Manager*, klik menu *Sessions* dan *Start-up*.
3. Pilih dan cek pilihan *Display chooser on login and close* semua menu.
4. *Log-out* dari sistem Linux Anda.
5. Kemudian akan ada pilihan menu. Pilih *new session*, dan beri nama sesuai keinginan Anda.
6. Klik new session yg tadi Anda buat.
7. Sekarang, taskbar Xfce Anda akan kembali seperti konfigurasi awal.

Mengaktifkan Autorecovery System

Pada distro ZenCafe, sudah disertakan sebuah *tool* bernama Autorecovery System yang dapat berfungsi untuk mengembalikan *setting*-an dan isi *home directory* pada kondisi awal. Hal ini cukup bermanfaat pada lingkungan warnet yang PC-nya biasa digunakan oleh beberapa user dalam setiap harinya. Selain itu, *tool* ini juga dapat menjaga agar tampilan desktop dan file yang ditambah oleh user warnet, akan secara otomatis terhapus setiap kali komputer *restart*. Untuk segera mengaktifkan tool Autorecovery System, Anda dapat melakukan langkah sebagai berikut:

1. Klik menu *System -> Zenpanel*.
2. Dari halaman Zenpanel, klik menu *Autorecovery Settings*.
3. Setelah tampil halaman Autorecovery System, terdapat tiga buah pilihan yakni, *[Re]Install, Remove*, dan *Forget It*. Pilih [Re]Install untuk menggunakan Autorecovery System.
4. Berikutnya, isikan nama home direktori yang ingin Anda *protect* dengan tool Autorecovery System ini. Sebagai contoh, jika user Anda bernama jasmine, maka isikan saja jasmine, atau */home/jasmine* pada kotak isian tersebut.
5. Fitur autorecovery sudah diaktifkan. Anda dapat mencoba dengan membuat sebuah folder dan isikan dengan sejumlah file dalam folder tersebut, kemudian restart PC Anda. Setelah login kembali, maka folder berisi sejumlah file tersebut, akan langsung dihapus oleh tool ini.

Demikian beberapa tip dan trik seputar distro Zencafe, yang sumbernya berdasar pada tutorial yang terdapat pada situs Zencafe di *http://linux.blogs.ie/*. Karena distro ini dibuat sebagai turunan distro Zenwalk, halaman situs wiki Zenwalk juga dapat dijadikan sebagai panduan yang bagus dalam mempelajari Zencafe. Selamat mencoba!

Supriyanto [supriyanto@infolinux.co.id]