Bahasa C dan Database PostgreSQL

Anda membangun program dengan bahasa C? Membutuhkan database yang *powerful*? Silakan gunakan PostgreSQL. Dengan memanfaatkan pustaka yang telah tersedia, kita dapat membangun aplikasi dengan bahasa C, yang terhubung ke database PostgreSQL.

Ketika Anda membutuhkan pustaka untuk melakukan koneksi ke database PostgreSQL dari program C yang Anda bangun, Anda tidak perlu mencari pustaka tersebut ke mana-mana. PostgreSQL telah datang di antaranya dengan pustaka libpq, yang bisa digunakan dengan mudah dan nyaman. Libpq datang dengan semua fungsi yang diperlukan untuk berkomunikasi dengan server database PostgreSQL.

Di dalam "Tutorial" ini, kita akan membahas beberapa contoh aplikasi yang bekerja dengan database PostgreSQL, mulai dari yang sangat sederhana sampai contoh untuk kebutuhan dunia nyata:

- Contoh melakukan koneksi ke database server.
- Melakukan query select sederhana.
- Menampilkan isi tabel.
- Melakukan query delete dan insert.
- Contoh aplikasi data pasien.

Versi PostgreSQL yang digunakan di dalam tulisan ini adalah PostgreSQL 8.2.1. Sebelum memulai, pastikan Anda telah menginstal paket develoment PostgreSQL (rujuklah kepada dokumentasi distro Anda). Untuk file header, kita akan memerlukan libpq-fe.h.

Semua pembahasan di tulisan ini dibangun di atas sistem Zenwalk Linux 4.2, namun seharusnya bisa digunakan pada sistem lainnya tanpa banyak perubahan. Sebelum mengikuti pembahasan, perlu dimaklumi bahwa, contoh-contoh yang ada merupakan contoh yang dapat dikembangkan sesuai kebutuhan Anda sendiri, sesuai dengan lisensi yang dipergunakan. Kami barangkali tidak membangun aplikasi yang berfungsi penuh, dengan segudang fitur, karena akan memakan sangat banyak tempat dan sekaligus keluar dari batasan materi.

Melakukan koneksi ke server

Untuk melakukan koneksi ke server, kita akan menggunakan fungsi PQconnectdb() yang akan membutuhkan satu parameter berupa connection string PostgreSQL. Apabila koneksi berhasil, maka akan dihasilkan *reference* ke struktur data PGconn dengan status koneksi CONNECTION_OK.

Di contoh ini, kita akan melakukan koneksi dengan connection string didapatkan dari argumen yang diberikan oleh user. Selanjutnya, kita akan mencoba melakukan koneksi. Apabila koneksi gagal, kita akan menampilkan pesan kesalahan. Apabila koneksi berhasil, maka kita akan menampilkan beberapa informasi koneksi.

Sebelum aplikasi selesai, kita akan melakukan cleanup dengan memanggil fungsi PQfinish().

Berikut ini adalah source code 1.c:

```
/*
 * (c) Nop, 2007, GPLv2.
 */

#include <stdio.h>
#include <libpq-fe.h>


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

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

        conn = PQconnectdb
(argv[1]);

        if (PQstatus (conn) !=
CONNECTION_OK)
        {

                fprintf (stderr,
"Kesalahan koneksi: %s\n",
PQerrorMessage (conn));
        }
        else
        {
                fprintf (stdout,
"connected:\n");
                fprintf (stdout,
"\tto host: %s\n", PQhost (conn));
                fprintf (stdout,
"\tto database: %s\n", PQdb
(conn));
                fprintf (stdout,
```

```
"\tas user: %s\n", PQuser (conn));

        }


        fprintf (stdout,
"disconnecting...");
        PQfinish (conn);
        fprintf (stdout, "done\
n");

        return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

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

Setelah itu, jalankanlah program 1 yang telah dihasilkan:

```
$ ./1
usage: ./1 <connection_string>
```

Contoh koneksi gagal:

```
$ ./1 'dbname=notfound'
Kesalahan koneksi: FATAL:
database "notfound" does not exist

disconnecting...done
```

Contoh koneksi berhasil 1:

```
$ ./1 ''
connected:
        to host: (null)
        to database: nop
        as user: nop
disconnecting...done
```

Contoh koneksi berhasil 2:

```
$ ./1 'host=localhost dbname=nop
user=nop'
connected:
        to host: localhost
        to database: nop
        as user: nop
disconnecting...done
```

Penjelasan tambahan source code:

- PQstatus() dapat digunakan untuk menguji status hasil koneksi.
- PQerrorMessage() dapat digunakan untuk mendapatkan pesan kesalahan.
- PQhost() dapat digunakan untuk mendapatkan informasi *host* dari koneksi.
- PQdb() dapat digunakan untuk mendapatkan informasi database dari koneksi.
- PQuser() dapat digunakan untuk mendapatkan informasi user dari koneksi.
- Perhatikanlah juga cara penggunaan PQconnectdb() dan PQfinish().

Mendapatkan versi PostgreSQL

Untuk mendapatkan versi PostgreSQL, kita akan memanggil fungsi version() yang telah tersedia. Pada contoh ini, kita akan memanggil fungsi tersebut, mendapatkan hasilnya, kemudian menampilkannya ke stdout.

Berikut ini adalah source code 2.c:

```
/*
 * (c) Nop, 2007, GPLv2.
 */

#include <stdio.h>
#include <libpq-fe.h>

#define MAX_LEN 255

int main(int argc, char * argv[])
{
        PGconn *conn;
        PGresult *res;
        char * query = calloc
(MAX_LEN, sizeof (char));

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

        conn = PQconnectdb
(argv[1]);

        if (PQstatus (conn) !=
CONNECTION_OK)
        {

                fprintf (stderr,
"Kesalahan koneksi: %s\n",
PQerrorMessage (conn));
        }
        else
        {
                sprintf (query,
"select version() as version");
                res = PQexec
(conn, query);
                if (PQresultStatus
(res) == PGRES_TUPLES_OK)
                {
                        fprintf
(stdout, "version: %s\n",
PQgetvalue(res, 0,0));
                }
        }


        PQfinish (conn);

        return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut ini:

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

Setelah itu, jalankanlah program 2 yang telah dihasilkan:

```
$ ./2 'user=nop'
version: PostgreSQL 8.2.1 on i686-
pc-linux-gnu, compiled by GCC gcc
(GCC) 3.4.6
```

Penjelasan source code:

- Selain struktur data PGconn (koneksi), kita juga menggunakan PGresult (hasil query/result query).
- Pertama-tama, kita akan membangun query berikut:
  `select version() as version`
- Setelah itu, kita akan melempar query ke fungsi PQexec() yang akan membutuhkan parameter berupa struktur data koneksi dan query string. Hasilnya akan diarahkan ke struktur data PGresult.
- Untuk query yang mengembalikan nilai (seperti select), maka kita akan memeriksa apakah status result bernilai PGRES_TUPLES_OK. Apabila benar, maka kita akan menampilkan informasi yang telah didapatkan. Perhatikan bahwa kita menggunakan fungsi PQresultStatus() untuk mendapatkan status result.
- Untuk mendapatkan nilai dari *result query*, kita akan menggunakan fungsi PQgetvalue(). Fungsi ini akan mem-

butuhkan tiga parameter: struktur data result query, baris dan kolom. Baris dan kolom dihitung mulai dari 0.

Menampilkan isi tabel

Contoh berikut ini merupakan pengembangan lebih lanjut dari contoh sebelumnya. Kita masih tetap akan bermain di perintah select, namun kita tidak hanya mengambil baris pertama saja, karena kita akan menampilkan isi sebuah tabel.

Sebagai catatan, nama database yang dipergunakan adalah test dan tabel yang dipergunakan adalah a.

Perintah untuk membuat database test:

```
$ createdb -e --owner=nop -Unop
test
CREATE DATABASE test OWNER nop;
CREATE DATABASE
```

Perintah untuk membuat tabel a:

```
test=# create table a(id serial,
nama varchar(128), alamat
varchar(255), primary key(id));
NOTICE:  CREATE TABLE will create
implicit sequence “a_id_seq” for
serial column “a.id”
NOTICE:  CREATE TABLE / PRIMARY
KEY will create implicit index
“a_pkey” for table “a”
CREATE TABLE
test=#
```

Perintah-perintah untuk mengisikan data ke table a:

```
test=# insert into a(nama, alamat)
values (‘andi’,’aceh’);
INSERT 0 1
test=# insert into a(nama, alamat)
values (‘budi’,’bandung’);
INSERT 0 1
test=# insert into a(nama, alamat)
values (‘cherry’,’cimanggis’);
INSERT 0 1
test=# insert into a(nama, alamat)
values (‘deni’,’denpasar’);
INSERT 0 1
test=# insert into a(nama, alamat)
values (‘edi’,’enrekang’);
INSERT 0 1
```

Berikut ini adalah source code 3.c

```
/*
 * (c) Nop, 2007, GPLv2.
 */

#include <stdio.h>
#include <libpq-fe.h>

#define MAX_LEN 255

int main(int argc, char * argv[])
{
        PGconn *conn;
        PGresult *res;
        char * query = calloc
(MAX_LEN, sizeof (char));
        int field_count;
        int rec_count;
        int i,j;

        if (argc != 2)
        {
                fprintf (stderr,
“usage: %s <connection_string>\n”,
argv[0]);
                return 1;
        }

        conn = PQconnectdb
(argv[1]);

        if (PQstatus (conn) !=
CONNECTION_OK)
        {

                fprintf (stderr,
“Kesalahan koneksi: %s\n”,
PQerrorMessage (conn));
        }
        else
        {
                sprintf (query,
“select nama,alamat from a”);
                res = PQexec
(conn, query);
                if (PQresultStatus
(res) == PGRES_TUPLES_OK)
                {
                        field_
count = PQnfields (res);
                        for (i=0;
i< field_count; i++)
                        {

 fprintf (stdout, “%-40s”, PQfname
(res, i));
                        }
                        fprintf
(stdout, “\n”);
                        rec_count
= PQntuples (res);
                        for (i=0;
i< rec_count; i++)
                        {

 for (j=0; j< field_count; j++)
                                {

       fprintf (stdout, “%-40s”,
PQgetvalue (res, i, j));
                                }

 fprintf (stdout, “\n”);
                        }

                        PQclear
(res);

                }
        }

        PQfinish (conn);

        return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

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

Setelah kompilasi dilakukan, kita bisa menjalankan program 3 yang dihasilkan. Pastikan kita mengikutkan informasi dbname=test di dalam parameter koneksi.

Contoh output:

```
$ ./3 ‘dbname=test user=nop’
nama                    alamat
andi                    aceh
budi                    bandung
cherry                  cimanggis
deni                    denpasar
edi                     enrekang
```

Penjelasan source code:

- Lihatlah juga penjelasan 2.c
- Untuk mendapatkan jumlah field, kita mempergunakan fungsi PQnfields() yang akan membutuhkan parameter berupa result query.
- Untuk mendapatkan nama field, melalui perulangan, kita akan mempergunakan

fungsi PQfname() yang akan membutuhkan dua parameter, yaitu result query dan nomor kolom. Nomor kolom dimulai dari 0.

- Untuk mendapatkan jumlah baris, kita mempergunakan fungsi PQntuples() yang akan membutuhkan parameter berupa result query.
- Dengan jumlah baris dan kolom telah diketahui, kita bisa melakukan perulangan untuk mendapatkan isi tabel menggunakan fungsi yang juga telah dibahas di 2.c, yaitu PQgetvalue().
- Apabila result query sudah tidak dipergunakan, lakukan cleanup dengan fungsi PQclear(). Jangan lupakan yang satu ini.

Menghapus tabel, menambahkan 100000 record

Di dalam contoh ini, kita akan memberikan perintah delete dan insert. Perintah delete kita gunakan untuk menghapus seluruh isi tabel. Setelah itu, kita mengisikan 100.000 record ke dalam tabel tersebut.

Sebagai catatan, database yang dipergunakan adalah database test dan tabel yang dipergunakan adalah tabel b.

Perintah untuk membuat tabel b:

```
# create table b(b1 serial, b2
varchar, primary key(b1));
NOTICE:  CREATE TABLE will create
implicit sequence “b_b1_seq” for
serial column “b.b1”
NOTICE:  CREATE TABLE / PRIMARY
KEY will create implicit index
“b_pkey” for table “b”
CREATE TABLE
```

Berikut ini adalah source code 4.c:

```
/*
 * (c) Nop, 2007, GPLv2.
 */

#include <stdio.h>
#include <libpq-fe.h>

#define MAX_LEN 255
#define ROW_COUNT 100000

int main(int argc, char * argv[])
{
        PGconn *conn;
        PGresult *res;
        char * query = calloc
(MAX_LEN, sizeof (char));
        int i;

        if (argc != 2)
        {
                fprintf (stderr,
“usage: %s <connection_string>\n”,
argv[0]);
                return 1;
        }

        conn = PQconnectdb
(argv[1]);

        if (PQstatus (conn) !=
CONNECTION_OK)
        {

                fprintf (stderr,
“Kesalahan koneksi: %s\n”,
PQerrorMessage (conn));
        }

        else
        {
                sprintf (query,
“delete from b”);
                res = PQexec
(conn, query);
                if (PQresultStatus
(res) == PGRES_COMMAND_OK)
                {
                        fprintf
(stdout, “done deleting content of
b.\n”);
                        PQclear
(res);
                }


                for (i=0; i<
ROW_COUNT; i++)
                {
                        sprintf
(query, “insert into b(b2) values
(‘row %d’)”, i);
                        res =
PQexec (conn, query);
                        if
(PQresultStatus (res) != PGRES_
COMMAND_OK)
                        {

 fprintf (stderr, “\tfailed on
row: %d\n”, i);
 exit (1);
                        }
                        PQclear
(res);


                }
                fprintf (stdout,
“done adding %d rows\n”, ROW_
COUNT);

        }


        PQfinish (conn);

        return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

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

Jalankanlah 4 dengan informasi dbname=test ditambahkan ke *connection string*.

Penjelasan source code:

- Untuk memberikan query, kita tetap menggunakan PQexec().
- Untuk memeriksa status result yang tidak mengembalikan data, menggunakan fungsi PQresultStatus() kita akan memeriksa nilai PGRES_COMMAND_OK.
- Jangan lupa berikan PQclear() apabila diperlukan.

Data pasien

Di aplikasi contoh ini, kita akan menyediakan fungsi untuk menambahkan data pasien dan melihat data seluruh pasien.

Tampilan menu utama:

```
DATA PASIEN
***********

a Tambah data
b Tampil data
x Keluar aplikasi
Pilihan Anda:
```

Berikut ini adalah source code 5.c:

```
/*
 * (c) Nop, 2007, GPLv2.
 */

#include <stdio.h>
```

```
#include <libpq-fe.h>

#define MAX_LEN 255

int main(int argc, char * argv[])
{
        PGconn *conn;
        PGresult *res;
        char * conninfo = calloc
(MAX_LEN, sizeof (char));
        char * query = calloc
(MAX_LEN, sizeof (char));
        char * nama = calloc
(MAX_LEN, sizeof (char));
        char * alamat = calloc
(MAX_LEN, sizeof (char));
        char * temp = calloc
(MAX_LEN, sizeof (char));
        int field_count;
        int rec_count;
        int c,menu;
        int i,j;

        strcpy (conninfo,
“dbname=test user=nop”);
        conn = PQconnectdb
(conninfo);

        if (PQstatus (conn) !=
CONNECTION_OK)
        {

                fprintf (stderr,
“Kesalahan koneksi: %s\n”,
PQerrorMessage (conn));
                exit (1);
        }


        while ( 1 )
        {
                fprintf(stdout,
“\n\n\nDATA PASIEN\n”);
                fprintf(stdout,
“***********\n\n”);
                fprintf(stdout, “a
Tambah data\n”);
                fprintf(stdout, “b
Tampil data\n”);
                fprintf(stdout, “x
Keluar aplikasi\n”);
                fprintf(stdout,
“Pilihan Anda: “);
                c = tolower(fgetc
(stdin));
                menu = c;

                while (c != ‘\n’
&& c != EOF) c = fgetc (stdin);

                if (menu == ‘a’)
                {

fprintf(stdout, “Tambah data\n”);

fprintf(stdout, “==========\n”);

fprintf(stdout, “Nama    : “);
                        fgets
(nama, MAX_LEN-1, stdin);

fprintf(stdout, “Alamat  : “);
                        fgets
(alamat, MAX_LEN-1, stdin);
                        sprintf
(query, “insert into pasien (nama,
alamat) values (‘%s’,’%s’)”, nama,
alamat);
                        res =
PQexec (conn, query);
                        PQclear
(res);
                }
                else

                if (menu == ‘b’)
                {

fprintf(stdout, “Tampil data\n”);

fprintf(stdout, “==========\n”);
                        sprintf
(query, “select nama,alamat from
pasien”);
                        res =
PQexec (conn, query);
                        field_
count = PQnfields (res);
                        for (i=0;
i< field_count; i++)
                        {

 fprintf (stdout, “%-40s”, PQfname
(res, i));
                        }
                        fprintf
(stdout, “\n”);

                        rec_count
= PQntuples (res);
                        for (i=0;
i< rec_count; i++)
                        {

 for (j=0; j< field_count; j++)
                                {

       strcpy (temp, PQgetvalue
(res, i, j));

      temp[strlen(temp)-1] = 0;

       fprintf (stdout, “%-40s”,
temp);
                                }

 fprintf (stdout, “\n”);
                        }

                        PQclear
(res);

                }
                else

                if (menu == ‘x’)
                {

fprintf(stdout, “Bye\n”);
                        break;

                }
        };

        PQfinish (conn);

        free (nama);
        free (alamat);
        free (query);
        free (conninfo);
        free (temp);

        return 0;
}
```

Lakukanlah kompilasi dengan memberikan perintah berikut:

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

Agar tampilan lebih bagus dan mudah digunakan, Anda bisa menggunakan pustaka ncurses (sudah pernah dibahas di *InfoLINUX*). Sampai di sini dulu pembahasan kita. Selamat mengembangkan!

Noprianto [noprianto@infolinux.co.id

IKLAN

Membangun Kamus Dua Bahasa

Bagian 1 dari 2 Tulisan

Di dunia komputer, program kamus dapat digunakan untuk membantu mempelajari bahasa asing, termasuk ketika kita mempelajari dokumentasi program tertentu. Selain menggunakan program kamus yang sudah tersedia, kita bisa pula membangun program kamus sendiri. "Tutorial" ini akan membahas pembuatan kamus secara tuntas.

Program kamus yang akan kita buat kali ini merupakan program kamus dua bahasa. Sebagai contoh adalah Bahasa Inggris–Indonesia. Agar lebih mantap, kita akan memisahkan database kamus dengan program kamusnya. Dengan demikian, program kamus yang akan kita buat tidak akan secara kaku menyebut dirinya program kamus bahasa Inggris–Indonesia ataupun bahasa lainnya, melainkan akan membaca metadata bahasa dari database kamus. Kondisi demikian memungkinkan kita memiliki banyak database kamus (misal: Inggris–Indonesia, Jerman–Indonesia, dan lain-lain) dan tetap menggunakan satu program pembaca.

Untuk saat ini, kemampuan penerjemahan kamus kita masih berbasis kata. Dengan demikian, user memasukkan kata yang ingin dicari padanannya dalam bahasa lain, dan output kamus adalah padanan kata yang diinginkan. Agar lebih mantap, karena satu kata terkadang memiliki hubungan dengan kata lain, kita bisa memiliki beberapa kata yang memiliki arti sama atau berhubungan dengan satu kata di bahasa lain.

Sebagai contoh, kata "kerja" di dalam bahasa Indonesia bisa memiliki padanan berikut di bahasa Inggris: *work, job, occupation, activity*, dan lain sebagainya. Oleh karena itu, pencarian work, job, occupation, dan lainnya akan menghasilkan padanan "kerja". Kita tidak perlu membuat pasangan kata tersebut satu demi satu, namun cukup mencantumkan referensi dari satu kata ke kata lainnya yang berhubungan. Di dalam contoh sebelumnya, kita tidak perlu membuat entri kerja=work, kerja=job, kerja=occupation, dan seterusnya, namun cukup dengan kerja=work, di mana work memiliki semacam referensi ke job, occupation, dan lainnya. Suatu hari, andaikata kata 'kerja' digantikan dengan kata lainnya, kita hanya cukup mengubah satu entri.

Terkadang, suatu kata juga memiliki informasi tambahan. Saat ini, sebuah informasi tambahan tersedia untuk setiap kata, di masing-masing bahasa. Informasi tambahan ini belum digunakan secara intensif, namun di masa depan, dapat digunakan di antaranya untuk informasi pengucapan, ataupun informasi lainnya. Dan, ini sepenuhnya tergantung pada penyedia database kamus.

Untuk format database, kita menyerahkan sepenuhnya kepada SQLite (*http://www.sqlite.org*), sebuah database *self-contained* dan *embeddable* yang luar biasa. Berikut ini adalah beberapa fitur SQLite:

- Transaksi ACID bahkan setelah system crash dan kegagalan *power*.
- *Zero-configuration*.
- Mengimplementasikan sebagian besar standar SQL92.
- Satu file tunggal untuk satu database.
- Mendukung ukuran database sampai 2 tebibyte (2^41 byte).
- Ukuran string dan BLOB sampai 2 gibibyte (2^31 byte).
- Kecil, hanya berukuran 250 KB (atau bahkan 150 KB dengan beberapa fitur tidak dimasukkan).
- *Source code* berada di dalam *public domain*.

Fitur-fitur selengkapnya, termasuk dokumentasi, bisa dibaca di website-nya. Versi SQLite yang kita gunakan adalah SQLite 3.x.

Bagaimana dengan *user interface* kamus itu sendiri? Di tulisan ini, kita akan mencoba untuk mengakomodasi kebutuhan berbagai kalangan user. Berikut ini adalah beberapa user interface yang kita sediakan:

- *Command Line Interface*, menggunakan *shell script*.
- Command Line Interface, menggunakan bahasa C.
- *Text User interface*, menggunakan shell script dan dialog.
- *Graphical User Interface*, menggunakan shell script dan Xdialog (satu *code-base* dengan text user interface dengan shell script).
- *Web-based*, menggunakan bahasa PHP.

Semua pembahasan di tulisan ini dibangun di atas sistem Zenwalk Linux 4.2, namun seharusnya bisa digunakan pada sistem lainnya tanpa banyak perubahan. Sebelum mengikuti pembahasan, perlu dimaklumi bahwa contoh-contoh yang ada merupakan contoh yang dapat dikembangkan sesuai kebutuhan Anda sendiri, sesuai dengan lisensi yang dipergunakan. Kami barangkali tidak membangun aplikasi yang berfungsi penuh, dengan segudang fitur, karena akan

memakan sangat banyak tempat dan sekaligus keluar dari batasan materi.

Pembahasan database

Di bagian ini, kita akan membahas inti kamus, yaitu database SQLite. Sebelum melanjutkan pembahasan, pastikan SQLite 3.x telah terinstal pada sistem Anda. Rujuklah kepada dokumentasi distro Anda untuk instalasi paket program SQLite. Sebelum men-*download* source atau binary yang tersedia pada *www.sqlite.org,* periksalah terlebih dahulu repository distro yang Anda pergunakan. Saat ini, cukup banyak distro telah memaketkan SQLite.

Jalankanlah perintah berikut untuk menguji ketersediaan SQLite:

```
$ sqlite3 --version
3.3.13
```

Setelah itu, kita bisa mulai merancang sebuah database bahasa, di mana di dalamnya terdapat dua tabel:

- Info
  - lang1_desc, bertipe string, berguna untuk deskripsi bahasa pertama, misal: English.
  - lang2_desc, bertipe string, berguna untuk deskripsi bahasa kedua, misal: Bahasa Indonesia.
  - version, bertipe string, berguna untuk versi struktur database, saat ini, kita sepakati versi 1.
  - extra, bertipe string, berguna untuk informasi tambahan. Saat ini, belum dipergunakan.
- Dictionary
  - lang1, bertipe string, menyimpan kata bahasa pertama, misal: work.
  - lang1_ext, bertipe string, menyimpan informasi tambahan kata bahasa pertama. Bisa dipergunakan untuk informasi pengucapan. Saat ini, belum dipergunakan secara intensif.
  - lang1_ref, bertipe string, menyimpan kata-kata yang berhubungan dengan kata bahasa pertama, misal: job. Apabila lang1_ref ini diisikan, maka padanan pada bahasa lainnya tidak diperlukan lagi (oleh karena itu, dikosongkan saja).
  - lang2, sama dengan lang1, namun untuk bahasa kedua.
  - lang2_ext, sama dengan lang1_ext, namun untuk bahasa kedua.
  - lang2_ref, sama dengan lang1_ref, namun untuk bahasa kedua.
  - extra, bertipe string, untuk informasi tambahan. Saat ini, belum dipergunakan.

Catatan:
Tabel info seharusnya hanya berisikan satu *record* saja.

Jalankanlah program sqlite3, diikuti nama file database. Untuk nama database, disarankan untuk mengikuti aturan ISO 3166 (ISO 3166-1-alpha-2), dipisahkan oleh karakter – untuk setiap kode (contoh: en-id). Kita sepakati juga untuk memberikan ekstensi .db untuk nama file database. Sebagai contoh:

```
$ sqlite3 en-id.db
SQLite version 3.3.13
Enter ".help" for instructions
sqlite>
```

Berikut ini adalah perintah SQL untuk membangun struktur tabel:

```
CREATE TABLE dictionary(lang1
string, lang1_ext string, lang1_
ref string, lang2 string, lang2_
ext string, lang2_ref string,
extra string);
```

```
CREATE TABLE info(lang1_desc
string, lang2_desc string, version
string,extra string);
```

Berikanlah perintah berikut untuk mengisikan metadata pada tabel info:

```
insert into info(lang1_desc,
lang2_desc, version) values
('English','Bahasa Indonesia',
"1.0");
```

Berikanlah perintah-perintah berikut untuk mengisikan beberapa baris padanan kata:

```
insert into dictionary(lang1,lang1_
ext,lang1_ref,lang2,lang2_ext,
lang2_ref) values('eat','','','mak
an','','');
```

```
insert into dictionary(lang1,lang1_
ext,lang1_ref,lang2,lang2_ext,
lang2_ref) values('work','','','ke
rja','','');
```

```
insert into dictionary(lang1,lang1_
ext,lang1_ref,lang2,lang2_ext,
lang2_ref) values('occupation','',
'work','','','');
```

```
insert into dictionary(lang1,lang1_
ext,lang1_ref,lang2,lang2_ext,
lang2_ref) values('job','','work',
'','','');
```

Berikan perintah berikut untuk keluar dari prompt sqlite:

```
sqlite> .exit
```

Perintah SQL yang digunakan pada program:

- Membaca deskripsi bahasa pertama: select lang1_desc from info.
- Membaca deskripsi bahasa kedua: select lang2_desc from info.
- Melihat jumlah entri kamus: select count(*) from dictionary.
- Mencari bahasa 1:
  - Mendapatkan ref: select lang1_ref from dictionary where lang1='<kata_yang_dicari>'.
  - Apabila ref kosong: select lang2 from dictionary where lang1='<kata_yang_dicari>'.
  - Apabila ref ditemukan: select lang2 from dictionary where lang1='<ref>'.
  - Kita juga akan membaca kolom lang2_ext.
- Mencari bahasa 2:
  - Mendapatkan ref: select lang2_ref from dictionary where lang2='<kata_yang_dicari>'.
  - Apabila ref kosong: select lang1 from dictionary where lang2='<kata_yang_dicari>'.
  - Apabila ref ditemukan: select lang1 from dictionary where lang2='<ref>'.
  - Kita juga akan membaca kolom lang1_ext.

Setelah database selesai dikerjakan, kita akan memasuki pembahasan pembuatan aplikasi kamus.

Interface CLI dengan shell script

Program kamus yang dibangun dengan shell script ini membutuhkan tiga parameter ketika dijalankan:

- File database kamus.
- Bahasa yang dicari (diisikan 1 atau 2,

dimana 1 adalah bahasa pertama dan 2 adalah bahasa kedua).

- Kata yang ingin dicari.

Berikut ini adalah source code file nkamus-cli.sh:

```
#!/bin/sh

#nkamus.sh v0.0.1
#front end nkamus using shell script
#interface: CLI

APP_NAME="nkamus"
APP_CMD="nkamus-cli.sh"
APP_VERSION="0.0.1"
APP_AUTHOR="Noprianto"
APP_LICENSE="GPL"
APP_WEBSITE="http://www.noprianto.com/code.php"

SQLITE_BIN="/usr/bin/sqlite3"
APP_DBFILE=""

usage()
{
        echo "$APP_NAME version $APP_VERSION"
        echo "(c) $APP_AUTHOR, $APP_LICENSE"
        echo "usage: $1 <database_file> <source_lang> <search_string>"
        echo "<source_lang> = 1 | 2"
}

if [ ! $# -eq 3 ]
then
        usage $0
        exit 1
fi

APP_DBFILE="$1"
APP_SRCLANG="$2"
APP_SEARCHSTR="$3"

if [ ! -r "$APP_DBFILE" ]
then
        usage
        echo
        echo "Cannot open $APP_DBFILE"
        exit 2
fi

if [ -z "$APP_SEARCHSTR" ]
then
        usage
        exit 3
fi

echo "$APP_NAME version $APP_VERSION"

LANG1=`$SQLITE_BIN $APP_DBFILE 'select lang1_desc from info'`
LANG2=`$SQLITE_BIN $APP_DBFILE 'select lang2_desc from info'`
ENTRY_COUNT=`$SQLITE_BIN $APP_DBFILE 'select count(*) from dictionary'`

echo "Using database file: $APP_DBFILE"
echo "Dictionary: $LANG1 <-> $LANG2"
echo "Contains $ENTRY_COUNT entries"

case "$APP_SRCLANG" in
        1) echo "Searching $LANG1"
           REF=`$SQLITE_BIN $APP_DBFILE "select lang1_ref from dictionary where lang1='$APP_SEARCHSTR'"`
           if [ -z "$REF" ]
           then
                RES=`$SQLITE_BIN $APP_DBFILE "select lang2 from dictionary where lang1='$APP_SEARCHSTR'"`
                RES2=`$SQLITE_BIN $APP_DBFILE "select lang2_ext from dictionary where lang1='$APP_SEARCHSTR'"`
           else
                RES=`$SQLITE_BIN $APP_DBFILE "select lang2 from dictionary where lang1='$REF'"`
                RES2=`$SQLITE_BIN $APP_DBFILE "select lang2_ext from dictionary where lang1='$REF'"`
           fi
                ;;
        2) echo "Searching $LANG2"
           REF=`$SQLITE_BIN $APP_DBFILE "select lang2_ref from dictionary where lang2='$APP_SEARCHSTR'"`
           if [ -z "$REF" ]
           then
                RES=`$SQLITE_BIN $APP_DBFILE "select lang1 from dictionary where lang2='$APP_SEARCHSTR'"`
                RES2=`$SQLITE_BIN $APP_DBFILE "select lang1_ext from dictionary where lang2='$APP_SEARCHSTR'"`
           else
                RES=`$SQLITE_BIN $APP_DBFILE "select lang1 from dictionary where lang2='$REF'"`
                RES2=`$SQLITE_BIN $APP_DBFILE "select lang1_ext from dictionary where lang2='$REF'"`
           fi
                ;;
esac

echo

if [ -z "$RES" ]
then
        echo "No result"
else
        echo "$APP_SEARCHSTR:"
        if [ ! -z "$REF" ]
        then
                echo "Ref: $REF"
        fi
        echo "$RES"
        echo "$RES2"
fi
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x nkamus-cli.sh
```

Contoh output:

```
$ ./nkamus-cli.sh en-id.db 1 work
nkamus version 0.0.1
Using database file: en-id.db
Dictionary: English <-> Bahasa Indonesia
Contains 4 entries
Searching English

work:
kerja

$ ./nkamus-cli.sh en-id.db 1 job
nkamus version 0.0.1
Using database file: en-id.db
```

```
Dictionary: English <-> Bahasa
Indonesia
Contains 4 entries
Searching English

job:
Ref: work
kerja
```

Penjelasan source code:

- Pada dasarnya, kita hanya mengambil output dari perintah SQL yang kita berikan melalui program sqlite3 (diberikan sebagai argumen program).
- Untuk perintah SQL, lihatlah kembali pada pembahasan sebelumnya tentang database, untuk informasi selengkapnya.

Interface CLI dengan bahasa C

Program yang kita bangun dengan C ini memiliki user interface yang sama persis dengan program yang dibangun dengan shell script. Lalu, apa yang istimewa? Apabila dibandingkan dengan menggunakan program time, untuk waktu yang diperlukan secara rata-rata, program kamus yang dibangun dengan C bisa berjalan lebih cepat 6 sampai 10 kali dibandingkan dengan yang dibangun dengan shell script. Selain itu, kita sudah tidak perlu lagi binary program sqlite3. Kita hanya menggunakan pustaka libsqlite3. so.

Berikut ini source code nkamus.c:

```
#include <stdio.h>
#include <sqlite3.h>
#include <string.h>
#include <stdlib.h>

#define APP_NAME "nkamus"
#define APP_VERSION "0.0.1"
#define APP_AUTHOR "Noprianto"
#define APP_LICENSE "GPL"
#define APP_WEBSITE "http://www.
noprianto.com/code.php"
#define MAX_LEN 200

void usage()
{
        fprintf (stderr, "%s
version %s\n", APP_NAME, APP_
VERSION);
        fprintf (stderr, "(c) %s,
%s\n", APP_AUTHOR, APP_LICENSE);
        fprintf (stderr, "usage:
nkamus <database_file> <source_
lang> <search_string>\n");
        fprintf (stderr, "<source_
lang> = 1 | 2\n");
}

static int cb_get_first_rec(void
*param, int argc, char *argv[],
char *az_col_name[])
{
        int i;
        strcpy(param, argv[0]);
        return 0;
}

int main(int argc, char *argv[])
{
        sqlite3 *db;
        char * error_msg = calloc
(MAX_LEN, sizeof (char));
        char * query = calloc
(MAX_LEN, sizeof (char));
        char * ref = calloc (MAX_
LEN, sizeof (char));
        char * str_res = calloc
(MAX_LEN, sizeof (char));
        char * str_res2 = calloc
(MAX_LEN, sizeof (char));
        char * lang1 = calloc
(MAX_LEN, sizeof (char));
        char * lang2 = calloc
(MAX_LEN, sizeof (char));
        int res;
        if (argc != 4)
        {
                usage();
                return 1;
        }

        if (strcmp(argv[3],"") ==
0)
        {
                usage();
                return 2;
        }
        res = sqlite3_open
(argv[1], &db);
        if (res != SQLITE_OK)
        {
                fprintf(stderr,
"Cannot open %s\n", argv[1]);
                return 3;
        }

        fprintf (stdout, "%s
version %s\n", APP_NAME, APP_
VERSION);
        fprintf (stdout, "Using
database file: %s\n", argv[1]);


        sqlite3_exec(db, "select
lang1_desc from info", cb_get_
first_rec, lang1, &error_msg);
        sqlite3_exec(db, "select
lang2_desc from info", cb_get_
first_rec, lang2, &error_msg);
        fprintf(stdout,
"Dictionary: %s <-> %s\n", lang1,
lang2);

        sqlite3_exec(db, "select
count(*) from dictionary", cb_get_
first_rec, str_res, &error_msg);
        fprintf(stdout, "Contains:
%s entries\n", str_res);

        strcpy(str_res, "");
        strcpy(str_res2, "");

        if (strcmp(argv[2],"1")
== 0)
        {
                fprintf(stdout,
"Searching %s\n", lang1);
                sprintf(query,
"select lang1_ref from dictionary
where lang1='%s'", argv[3]);
                sqlite3_exec(db,
query, cb_get_first_rec, ref,
&error_msg);
                if (strlen(ref) <
1)
                {

sprintf(query, "select lang2 from
dictionary where lang1='%s'",
argv[3]);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res, &error_msg);

sprintf(query, "select lang2_ext
from dictionary where lang1='%s'",
argv[3]);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res2, &error_msg);

                }
                else
```

```
                {

sprintf(query, “select lang2 from
dictionary where lang1=’%s’”,
ref);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res, &error_msg);

sprintf(query, “select lang2_ext
from dictionary where lang1=’%s’”,
ref);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res2, &error_msg);
                }
        }

        else if
(strcmp(argv[2],”2”) == 0)
        {
                fprintf(stdout,
“Searching %s\n”, lang2);
                sprintf(query,
“select lang2_ref from dictionary
where lang2=’%s’”, argv[3]);
                sqlite3_exec(db,
query, cb_get_first_rec, ref,
&error_msg);
                if (strlen(ref) <
1)
                {

sprintf(query, “select lang1 from
dictionary where lang2=’%s’”,
argv[3]);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res, &error_msg);

sprintf(query, “select lang1_ext
from dictionary where lang2=’%s’”,
argv[3]);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res2, &error_msg);
                }
                else
                {

sprintf(query, “select lang1 from
dictionary where lang2=’%s’”,
ref);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res, &error_msg);

sprintf(query, “select lang1_ext
from dictionary where lang2=’%s’”,
ref);
                        sqlite3_
exec(db, query, cb_get_first_rec,
str_res2, &error_msg);
                }
        }
        fprintf(stdout, “\n”);
        if (strlen (str_res) < 1)
        {
                fprintf (stdout,
“No Result\n”);
        }
        else
        {
                fprintf (stdout,
“%s:\n”, argv[3]);
                if (strlen(ref) >
0)
                {
                        fprintf
(stdout, “Ref: %s\n”, ref);
                }
                fprintf (stdout,
“%s\n”, str_res);
                fprintf (stdout,
“%s\n”, str_res2);

        }

        free (query);
        free (ref);
        free (str_res);
        free (str_res2);
        free (lang1);
        free (lang2);
        sqlite3_free (error_msg);
        sqlite3_close (db);
        return 0;
}
```

Untuk kompilasi, berikanlah perintah berikut ini:

```
$ gcc -o nkamus nkamus.c -lsqlite3
```

Setelah itu, program nkamus akan dihasilkan. Cara penggunaanya sama persis dengan program yang dibangun dengan shell script sebelumnya.

Penjelasan source code:

- Yang pertama-tama, kita akan menggunakan header sqlite3.h.
- Untuk membuka database file, kita menggunakan fungsi sqlite3_open()
- Untuk memberikan perintah SQL, kita menggunakan fungsi sqlite3_exec(). Untuk kebutuhan mendapatkan record pertama, kita telah menyiapkan sebuah fungsi callback dengan nama cb_get_first_rec().
- Untuk menutup database, gunakanlah fungsi sqlite3_close().
- Untuk dokumentasi selengkapnya, bacalah dokumentasi di *www.sqlite.org*.

Interface TUI dengan shell script dan dialog

User yang menggunakan program ini tidak perlu lagi memberikan argumen apapun ketika menjalankan program. Melainkan, user akan dipandu menggunakan dialog-dialog yang mudah digunakan. Agar program dapat dijalankan dengan benar, pastikan program dialog telah terinstal di sistem Anda.

Berikut ini adalah source code nkamus-tui.sh

```
#!/bin/sh

#nkamus.sh v0.0.1
#front end nkamus using shell
script
#interface: TUI using dialog

APP_NAME=”nkamus”
APP_CMD=”nkamus-tui.sh”
APP_VERSION=”0.0.1”
APP_AUTHOR=”Noprianto”
APP_LICENSE=”GPL”
APP_WEBSITE=”http://www.noprianto.
com/code.php”
SQLITE_BIN=”/usr/bin/sqlite3”

APP_DBFILE=””
DIALOG=/bin/dialog
alias adialog=”$DIALOG --backtitle
‘$APP_NAME version $APP_VERSION,
(c) $APP_AUTHOR, $APP_LICENSE’”

TEMP=/tmp/nkamus.tmp

adialog --fselect `pwd`  10 40
2>$TEMP
[ $? -ne 0 ] && exit -1
APP_DBFILE=`cat $TEMP`
if [ ! -r “$APP_DBFILE” ]
then
        adialog --msgbox “Cannot
open $APP_DBFILE” 10 40
```

```
        exit 2
fi

LANG1=`$SQLITE_BIN $APP_DBFILE
'select lang1_desc from info'`
LANG2=`$SQLITE_BIN $APP_DBFILE
'select lang2_desc from info'`
ENTRY_COUNT=`$SQLITE_BIN $APP_
DBFILE 'select count(*) from
dictionary'`

while [ 1 ]
do
        adialog --menu  "$LANG1 <-
> $LANG2 ($ENTRY_COUNT entries)"\
                10 60 4 1 "$LANG1
-> $LANG2" 2 "$LANG2 -> $LANG1" 2>
$TEMP
        [ $? -ne 0 ] && break
        APP_SRCLANG=`cat $TEMP`

        while [ 1 ]
        do
                adialog --inputbox
"search string" 10 40 2>$TEMP
                [ $? -ne 0 ] &&
break
                APP_SEARCHSTR=`cat
$TEMP`
                [ -z "$APP_
SEARCHSTR" ] && continue
                case "$APP_
SRCLANG" in
                        1)
REF=`$SQLITE_BIN $APP_DBFILE
"select lang1_ref from dictionary
where lang1='$APP_SEARCHSTR'"`
                                if [ -z
"$REF" ]
                                then

  RES=`$SQLITE_BIN $APP_DBFILE
"select lang2 from dictionary
where lang1='$APP_SEARCHSTR'"`

 RES2=`$SQLITE_BIN $APP_DBFILE
"select lang2_ext from dictionary
where lang1='$APP_SEARCHSTR'"`
                                else

  RES=`$SQLITE_BIN $APP_DBFILE
"select lang2 from dictionary
where lang1='$REF'"`

 RES2=`$SQLITE_BIN $APP_DBFILE
"select lang2_ext from dictionary
where lang1='$REF'"`
                                fi
                                ;;
                        2)
REF=`$SQLITE_BIN $APP_DBFILE
"select lang2_ref from dictionary
where lang2='$APP_SEARCHSTR'"`
                                if [ -z
"$REF" ]
                                then

  RES=`$SQLITE_BIN $APP_DBFILE
"select lang1 from dictionary
where lang2='$APP_SEARCHSTR'"`

 RES2=`$SQLITE_BIN $APP_DBFILE
"select lang1_ext from dictionary
where lang2='$APP_SEARCHSTR'"`
                                else

  RES=`$SQLITE_BIN $APP_DBFILE
"select lang1 from dictionary
where lang2='$REF'"`

 RES2=`$SQLITE_BIN $APP_DBFILE
"select lang1_ext from dictionary
where lang2='$REF'"`
                                fi
                                ;;
                esac

                if [ -z "$RES" ]
                then
                        adialog --
msgbox  "No result" 10 40
                else
                        RES_STR=""
                        if [ ! -z
"$REF" ]
                        then

 RES_STR="Ref: $REF"
                        fi
                        RES_
STR="$RES_STR \n $RES"
                        RES_
STR="$RES_STR \n $RES2"
                        adialog
--title "$APP_SEARCHSTR:" --msgbox
"$RES_STR" 10 40
                fi
        done
done
unalias adialog
rm -f $TEMP
```

Berikanlah hak akses executable dengan perintah berikut:

```
$ chmod +x nkamus-tui.sh
```

Selanjutnya, user bisa menjalankan program kamus dengan memberikan perintah:

```
$ ./nkamus-tui.sh
```

Penjelasan source code:

- Logika program sama persis dengan program-program sebelumnya.
- Hanya, beberapa perintah tambahan diberikan untuk mengantisipasi input kosong yang diberikan, ataupun penekanan tombol *cancel*.

Interface GUI dengan shell script dan Xdialog

Bagi user yang lebih suka interface grafikal, kita akan menyediakannya juga. Agar tetap dapat memanfaatkan program yang dibangun dengan bantuan dialog, dan tetap menggunakan code base yang sama, kita akan menggunakan bantuan Xdialog untuk dialog grafikal. Pastikan Xdialog terinstal pada sistem Anda.

Dengan demikian, kita bisa meng-copy-kan file nkamus-tui.sh ke nkamus-gui.sh, kemudian melakukan beberapa perubahan berikut di file nkamus-gui.sh:

Mengubah variabel DIALOG, dari /bin/dialog sebelumnya ke /usr/bin/Xdialog.

Setiap definisi *height* dan *width* di setiap dialog, masing-masing height dan width kita kalikan dengan 5.

Contoh di nkamus-tui.sh:

```
adialog --inputbox "search string"
10 40 2>$TEMP
```

Contoh di nkamus-gui.sh:

Setelah height dan width masing-masing dikalikan dengan 5, menjadi:

```
adialog --inputbox "search string"
50 200 2>$TEMP
```

Selanjutnya, cara penggunaannya sama persis.

Untuk interface berbasis web, karena keterbatasan tempat, kita akan membahasnya pada edisi berikut. Sampai di sini dulu pembahasan kita. Selamat mengembangkan!

Noprianto [noprianto@infolinux.co.id

Simulator Jaringan Komputer NS2

Jaringan kabel maupun nirkabel dapat disimulasikan dalam NS2. Sebagian besar orang menggunakan NS2 dalam penelitian, namun tidak menutup kemungkinan digunakan untuk menguji trafik jaringan. Anda dapat belajar jaringan tanpa harus memasang kabel atau peralatan *wireless*.

Struktur NS2 (Network Simulator versi 2) seperti terlihat dalam Gambar 1.

Otcl merupakan kepanjangan dari MIT Object TCL, sebagai perluasan dari Tcl/Tk untuk pemrograman berbasis objek. tclcl berfungsi menghubungkan C++ dengan otcl. Implementasi NS2 terlihat dalam Gambar 2.

Pada Gambar 2 terdapat dua bahasa yang digunakan oleh NS2, yaitu bahasa C++ dan Tcl. Mengapa NS-2 menggunakan dua bahasa?

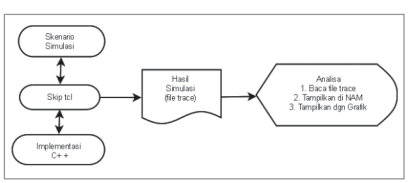
Berikut ini beberapa kelebihan dan kekurangan bahasa C++:

- Manipulasi byte, pemrosesan paket, dan implementasi algoritma.
- Kecepatan *running* sangat penting.
- C++ cepat namun lebih lama untuk mengganti kode.

Berikut ini kelebihan dan kekurangan bahasa Tcl:

- Cepat membuat berbagai skenario, di mana kita dapat membuat simulasi. dalam berbagai macam parameter dan konfigurasi.
- Tcl mudah dalam penggantian kode program, tapi berjalan lambat.
- Tcl secara periodik dapat mengontrol dan membangkitkan sebuah aksi.

Gambar 1. Struktur NS2.



Gambar 2. Implementasi NS2.

Beberapa fitur dalam NS2 adalah:

1. TCP dan implementasinya.
2. Mobilitas.
3. Node berupa satelit yang dapat kita tentukan *longtitude, latitude,* dan *altitude*.
4. Internet routing protocol.
5. Berbagai macam model *link loss*.
6. Berbagai macam penghasil *traffic* (contoh: web, telnet, dan sebagainya).
7. Kemampuan mengemulasikan jaringan.

Banyak fitur baru NS2, yang biasanya perlu tambahan modul.

Instalasi NS2

1. Download NS2 dari *http://sourceforge.net/*, atau Anda dapat menemukannya di dalam DVD *InfoLINUX* edisi ini. Pastikan juga, kalau di sistem operasi sudah memiliki Tcl, Tk, Otcl, dan Tclcl.
2. Ekstrak file ns-allinone-2.30.tar.gz

```
# tar zxvf ns-allinone-2.30.
tar.gz
```

   Struktur direktori NS2 terlihat seperti dalam Gambar 3.
3. Jalankan instalasi dengan lebih dulu masuk ke direktori hasil ekstrak, lalu jalankan perintah ./install.

```
# cd ns-allinone-2.30
# ./instal
```

4. Jika tidak ada kesalahan dan instalasi berjalan lancar, *copy*-kan isi folder bin pada /usr/sbin :

```
# cp bin/* /usr/sbin
```

5. Cek instalasi dengan dengan lebih dulu masuk direktori ns-2.30/tcl/ex, lalu jalankan perintah "ns simple.tcl".

```
# cd ns-2.30/tcl/ex/
# ns simple.tcl
```

   Pada konsole akan terdapat tulisan:

```
210
0.003749999999999999
running nam...
```

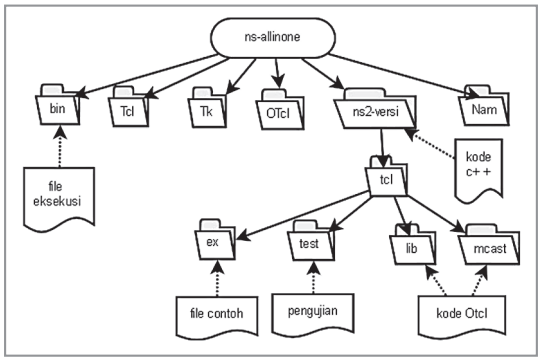
   Dan akan tampil GUI NAM seperti terlihat dalam Gambar 4.
6. Jika kita tekan tombol *play*, maka kita akan mengetahui bagaimana simulasi jaringan di antara 4 node tersebut.

Membuat simulasi dalam NS2

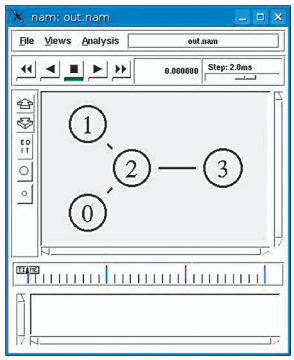
Untuk membuat simulasi kita perlu mengetahui bahasa Tcl. Bahasa Tcl tidak terlalu sulit, tapi salah *line break* bisa membuat *syntax error*. Berikut ini sedikit contoh penggunaan bahasa Tcl:]:

1. Komentar menggunakan tanda pagar (#), namun jika satu baris dengan perintah tambahkan tanda titik koma (;) di depan. Contoh:

```
#ini komentar
set q 3    ;#komentar dan
perintah
```



Gambar 3. Struktur Direktori NS-2.



Gambar 4. Tampilan NAM.

2. Variabel tidak perlu dideklarasikan, langsung pakai. Contoh set b 1 sebagai pengganti b=1.
3. Ekspresi harus eksplisit: set a [expr 3+4]
4. Menggunakan $ untuk membaca variabel : set a $b.
5. Menampilkan output: puts "hallo".
6. Objek pada Otcl diperlukan seperti variabel. Cara pemanggilan metode pada objek Otcl: "$nama objek metode parameter1 parameter2".

Urutan pembuatan simulasi:

1. Pembuatan simulator baru (new Simulator).
2. Skrip berisi skenario dan penjadwalan setiap kejadian. Untuk penjadwalan ini dapat digunakan kata kunci at waktu perintah.
3. Di akhir skrip ada $ns run.
4. Ns akan menghasilkan file output jika kita buat.
5. Simulasi akan terlihat interaktif jika kita menampilkannya pada network animator(NAM). Namun file nam ini sangat besar.
6. Hasil simulasi dapat ditampilkan dalam xgraph.

Berikut ini contoh sederhana skrip Tcl tanpa ada kaitannya dengan simulasi jaringan.

```
# Skrip Tcl sederhana
set ns [new Simulator]       ;#
membuat simulator baru
$ns at 1 "puts \"INFOLINUX\""
;#tampilkan tulisan INFOLINUX pada
detik ke-1

proc coba {} {          ; #membuat
prosedur coba
set a 3           ; #a=3
for {set k 0} {$k < 10} {incr k} {
; #for (k=0;k<10;k++)
   set b [expr $a + $k]    ;
#b=a+k;
   puts "a = $b"
}
}

$ns at 1.1 "coba"          ;#
jalankan prosedur coba pada detik
ke-1.1
$ns at 1.5 "exit"          ;#
keluar dari aplikasi pada detik
ke-1.5
$ns run           ;# jalankan
simulasi
```

Kode dasar pembuatan simulator jaringan pada skrip Tcl adalah:

```
#membuat objek simulator
set ns [new Simulator]

#membuat file trace bernama out.
nam yang akan ditampilkan pada nam
set nf [open out.nam w]
#semua kejadian dicatat dalam nf
$ns namtrace-all $nf

#deklarasikan proses penyelesaian
simulasi dengan menutup file trace
dan memulai Nam
proc selesai {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
#perintah finish akan dijalankan 5
detik setelah simulasi.
$ns at 5.0 "selesai"

#jalankan simulasi
$ns run
```

Kode di atas belum terdapat isi jaringannya. Berikut ini kita mulai pembuatan simulasi jaringan.

1. Membuat Node:

```
set n0 [$ns node]
```

Node adalah anggota dari class Simulator. Jika node bukan router kita perlu mendefinisikan traffic agents (TCP, UDP, dan lain-lain) dan traffic source (FTP, CBR, dan lain-lain).

Menghubungkan antar-node dapat mengunakan *simplex link* (satu arah) atau *duplex link* (dua arah). Hubungan antara dua node membutuhkan parameter *bandwitdh*, waktu *delay* dan jenis antrian paket. Jenis antrian paket yang paling sederhana adalah DropTail di mana mengantrikan paket berdasarkan waktu kedatanganya(FIFO). Contoh:

```
$ns simplex-link $n0 $n1 1Mb
10ms DropTail
$ns duplex-link $n0 $n1 1Mb 10ms
DropTail
```

2. Membuat Agent, aplikasi, dan sumber trafik.

Agent yang sering digunakan pada NS-2 adalah UDP dan TCP. Sedangkan untuk aplikasi dan sumber trafik yang sering digunakan adalah:

- Application/FTP: menghasilkan data yang akan dikirim oleh TCP.
- Application/Traffic/CBR: menghasilkan paket dengan nilai bit konstan.
- Application/Traffic/Trace: Trafik dihasilkan dari file trace di mana ukuran dan waktu pengiriman sudah ditentukan.

Cara membuat Agent:

- Membuat agent UDP

```
set udp0 [new Agent/UDP]
```

- Meletakkan UDP pada node0

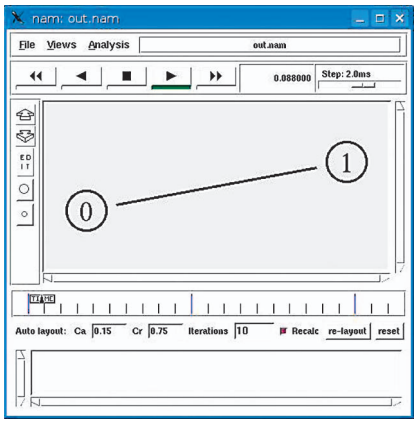
```
$ns attach-agent $n0 $udp0
```

- Cara membuat aplikasi

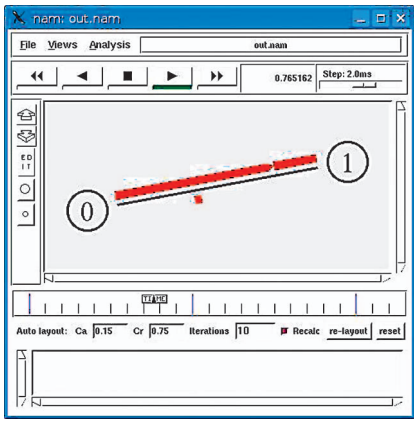
```
set ftp [new Application/FTP]
$ftp set type_ FTP
```

- Meletakkan ftp pada agent tcp

```
$ftp attach-agent $tcp
```



Gambar 5. Sebelum detik ke 0.5.



Gambar 6. Setelah detik ke 1.

Berikut ini contoh program Tcl sederhana dalam simulator jaringan:

```
#file contoh.tcl
#membuat simulator baru
set ns [new Simulator]

#definisi warna merah
$ns color 0 red

#membuat file untuk dijalankan
pada NAM
set nf [open tampilan.nam w]
$ns namtrace-all $nf

#membuat prosedur selesai
proc selesai {} {
        global ns nf
        $ns flush-trace ;# menutup
file trace
   close $nf
   exec nam out.nam & ;#Ekesekusi
file nam setelah selesai simulasi
        exit 0
}

#Membuat 2 node
set n0 [$ns node]
set n1 [$ns node]

#Membuat duplex link antara node 1
dan 2 dengan antrian droptail
$ns duplex-link $n0 $n1 1Mb 5ms
DropTail ;# bandwith 1Mb dengan
kecepatan 5ms

#Membuat agen TCP,
TCPSink(penerima TCP) dan FTP
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
set ftp [new Application/FTP]
$ns attach-agent $n0 $tcp ;#
letakkan tcp pada node 0
$ns attach-agent $n1 $sink;#
letakkan sink pada node 1
$ns connect $tcp $sink    ;#
hubungkan trafik node0(tcp) dan
node1(sink)
$ftp attach-agent $tcp    ;#jenis
TCP yang dikirim adalah ftp
$tcp set fid_ 0      ;#warna paket
merah pd NAM

#penjadwalan
$ns at 0.5 "$ftp start"    ;#detik
ke 0.5 ftp paket dikirim
$ns at 1.0 "$ftp stop"     ;#detik
ke 1.0 ftp paket dihentikan

# memanggil prosedur selesai pada
detk ke 1.5
$ns at 1.5 "selesai"
# jalankan simulasi
$ns run
```

Hasil dari skrip di atas, setelah dijalankan dengan perintah "#ns contoh.tcl", akan terlihat seperti pada Gambar 5 dan 6.

Tampilan node pada NAM dapat kita ubah melalu skrip. Skrip tersebut antara lain dapat menentukan warna, label, dan bentuk.

1. Warna.
   ```
   $nama_variabel_node color warna
   ```
   Contoh:
   ```
   $n0 color blue
   $n1 color chocolate
   ```

2. Label.
   ```
   $nama_variabel_node label
   "keterangan"
   ```
   Contoh:
   ```
   $n0 label "ftp server"
   $n1 label "ftp client"
   ```

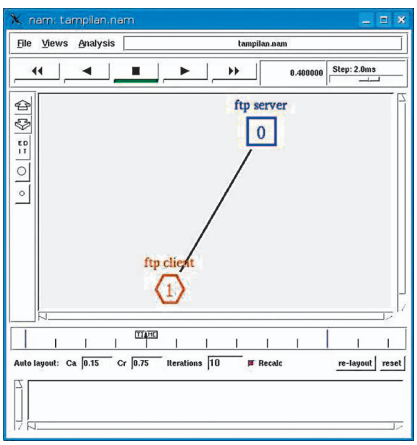
3. Bentuk pada Node ada tiga jenis, yaitu *circle*, *box*, dan *hexagon*.
   ```
   $nama_variabel_node shape
   jenis_bentuk
   ```
   Contoh:
   ```
   $n0 shape box
   $n1 shape hexagon
   ```

Untuk bentuk node tidak dapat kita ubah-ubah selama simulasi berlangsung, namun warna dan label dapat kita ganti. Contoh:

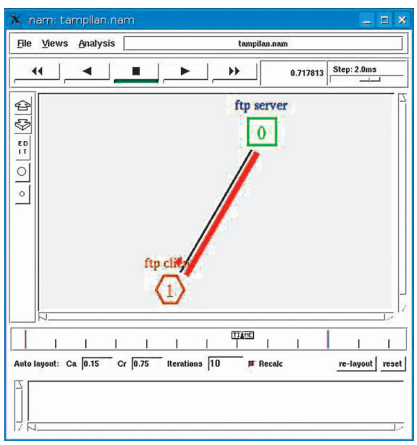
```
$ns at 0.7 "$n0 color green"
```

Jika kita tambahkan kode-kode di atas, dan di jalankan lagi #ns contoh.tcl maka akan terlihat Gambar 7 yang menunjukkan tampilan sebelum detik ke 0, 7, dan 8 yang menunjukkan tampilan sesudah detik ke 0,7.

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



Gambar 8. Setelah detik ke 0.71



Gambar 7. Pada saat detik ke 0.4

Koneksi VPN di Linux untuk e-Banking

Artikel ini bukan iklan. Penulis membagi pengalaman kepada Anda cara menggunakan koneksi VPN di Linux, dengan studi kasus aplikasi *Internet banking* dari salah satu bank swasta. Anda dapat menggunakan Linux, meskipun bank hanya menyediakan *software* VPN Client untuk Windows.

Pernahkah Anda mendengar "KlikBCA Bisnis"? Itu merupakan aplikasi internet banking untuk pengusaha atau perusahaan bagi pelanggan bank BCA. Selain koneksi internet, aplikasi ini menggunakan dalam VPN dan PIN generator.

Jika Anda menggunakan layanan ini, akan memperoleh KeyBCA (PIN generator) dan VPN client (berbasis Windows). Bagaimanakah dengan pengguna Linux? Pertanyaan ini pernah penulis lontarkan ke Customer Service KlikBCA Bisnis, dan mereka menjawab, "Selain Windows, belum ada".

Dengan diundangkannya UU No. 19 Tahun 2002 tentang Hak Cipta, diperkirakan banyak perusahaan yang beralih ke OS Linux. Adanya keterbatasan support dari KlikBCA Bisnis terhadap pengguna Linux, penulis mencari solusi ini dengan bantuan mesin pencari Internet, Google.

Untuk koneksi ke KlikBCA Bisnis ini, penulis menggunakan Cisco VPN Client versi 4.8 yang di-*download* dari Universitiet Gent: *http://helpdesk.ugent.be/vpn/en/linux.php*.

Dasar dari penggunaan "Cisco VPN Client for Linux" ini disebabkan oleh isi CD/disk kit yang dibagikan BCA untuk pengguna KlikBCA Bisnis, terdapat instalasi VPN Client for Windows dari Cisco. Dengan vendor yang sama, kita tidak bingung dengan konfigurasi koneksi VPN.

Saat instalasi, penulis menggunakan Ubuntu 6.10 karena cukup simpel untuk user biasa (bukan orang TI). Ubuntu termasuk distro yang populer. Selain cepat dan mudah dalam instalasi, dukungan *update*-nya cukup banyak di server-server lokal Indonesia.

Sebelumnya kita perlu mempersiapkan PC yang sudah terinstal Linux dan file profile tentang KlikBCA Bisnis (diambil dari PC yang terinstall Cisco VPN client for Windows). File profile ini sangat kita butuhkan dalam instalasi, karena berisi konfigurasi tunnel, IP server, *username*, dan *password* (terenkripsi) untuk terhubung ke VPN server milik KlikBCA. VPN server ini semacam dialup server di dalam media *network*/Internet. Oleh BCA, Username dan password di sini tidak diberikan ke user tetapi terinstal secara *default*, karena hanya digunakan sebagai koneksi/tunnel.

Dalam koneksi VPN dari Cisco ini, ada dua macam otentikasi, yaitu otentikasi untuk koneksi dan otentikasi untuk User. Otentikasi untuk User diberikan BCA dengan passwordnya dari KeyBCA. Otentikasi untuk koneksi tersimpan dalam file profile "KlikBCA Bisnis.pcf".

Instalasi

- Sebaiknya disediakan satu komputer khusus untuk client KlikBCA Bisnis ini. Dalam arti, tidak digunakan untuk file/printer sharing, internet server, atau client email. VPN dari Cisco ini akan memutuskan koneksi dengan LAN saat login sukses. IP untuk komputer ini boleh private, tetapi harus dapat terhubung secara langsung (via NAT) ke server klikBCA. Jika sering mencetak transkrip transaksi, sebaiknya disediakan printer yang terhubung langsung ke komputer. Bisa juga disimpan/dicetak dulu ke file, di simpan, dan dicetak ke printer sharing LAN setelah koneksi VPN selesai.
- Copy file KlikBCA Bisnis.pcf dari komputer Windows yang sudah terinstal ke dalam disket atau CD. Untuk memudahkan penggunaan, ubah nama (*rename*) file di dalam disket/CD tersebut menjadi klikbca.pcf (huruf kecil semua, tanpa spasi).
- Langkah kedua, copy file vpnclient-linux-x86_64-4.8.00.0490-k9.tar.gz ke folder home di Ubuntu, contoh /home/aku. Setelah itu, exctract. Jika menggunakan Ubuntu, ada Archive Manager yang siap membantu. Contoh menggunakan terminal, ketik:

```
$ cd /home/aku
$ tar xvzf vpnclient-linux-x86_
64-4.8.00-0490-k9.tar.gz
```

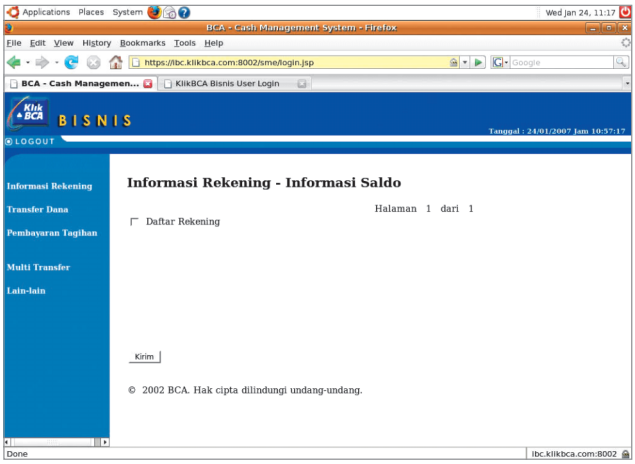
- Masuk ke folder yang sudah terbuat, lalu sebagai root jalankan perintah vpn_install. Jika perintah sudo tidak ada atau belum di-*setup*, Anda dapat menjalankan perintah su lebih dahulu dengan syarat Anda sudah membuat password root sebelumnya.

```
$ cd vpnclient
$ sudo ./vpn_install
```

- Kemudian akan muncul dialog sebagai berikut:

```
Cisco Systems VPN Client Version
```

Layar di browser jika VPN sukses terhubung.



Tampilan setelah login di KlikBCA Bisnis.

```
4.8.00 (0490) Linux Installer
Copyright (C) 1998-2005 Cisco
Systems, Inc. All Rights
Reserved.

By installing this product you
agree that you have read the
license.txt file (The VPN Client
license) and will comply with
its terms.


Directory where binaries will be
installed [/usr/local/bin]

Automatically start the VPN
service at boot time [yes]

In order to build the VPN kernel
module, you must have the
kernel headers for the version of
the kernel you are running.


Directory containing linux kernel
source code [/lib/modules/2.6.17-
10-generic/build]

* Binaries will be installed in
"/usr/local/bin".
* Modules will be installed in
"/lib/modules/2.6.17-10-generic/
CiscoVPN".
* The VPN service will be started
AUTOMATICALLY at boot time.
* Kernel source from "/lib/
modules/2.6.17-10-generic/build"
will be used to build the module.

Is the above correct [y]
```

- Setelah menekan Enter, proses kompilasi akan berjalan, dan hasilnya akan muncul sebagai berikut:

```
Setting permissions.
    /opt/cisco-vpnclient/bin/
cvpnd (setuid root)
    /opt/cisco-vpnclient (group
bin readable)
    /etc/opt/cisco-vpnclient
(permissions not changed)
* You may wish to change these
permissions to restrict access
to root.
* You must run "/etc/init.
d/vpnclient_init start" before
using the client.
* This script will be run
AUTOMATICALLY every time you
reboot your computer.
```

Catatan: vpnclient_init merupakan "driver" yang dibutuhkan untuk menjalankan vpnclient. vpnclient_init akan dijalankan saat komputer menyala kali pertama (*booting*). Anda tidak perlu melakukan reboot untuk menjalankan vpnclient setelah install ini, tetapi cukup ketik perintah sudo /etc/init.d/vpnclient_init start di terminal.

- Langkah berikutnya, copy file klikbca.pcf ke folder /etc/opt/vpnclient/Profile.
- Sampai langkah ini, KlikBCA Bisnis siap dijalankan.

Penggunaan

- Buka terminal dan jalankan perintah berikut ini:

```
$ sudo vpnclient connect klikbca
Password:

Cisco Systems VPN Client Version
4.8.00 (0490)
Copyright (C) 1998-2005 Cisco
Systems, Inc. All Rights
Reserved.
Client Type(s): Linux
Running on: Linux 2.6.17-10-
generic #2 SMP Tue Dec 5 22:28:26
UTC 2006 i686
Config file directory: /etc/opt/
cisco-vpnclient

Initializing the VPN connection.
Initiating TCP to 202.6.211.33,
port 10000
Contacting the gateway at
202.6.211.33
User Authentication for
klikbca...

The server has requested the
following information to complete
the user authentication:

Username []:
Password :
```

Catatan:

- Setelah kita ketik perintah di atas, akan ada pertanyaan password. Password ini untuk menjalankan perintah setingkat root. Untuk menghilangkan pertanyaan ini, kita akan bahas selanjutnya.
- Username yang ditanyakan adalah username sesuai yang diberikan dari BCA, dengan format: [namaperusahaan][id pengguna].

- Password yang ditanyakan (di baris terbawah), merupakan hasil "generate" dari KeyBCA.
- Setelah login sukses, komputer yang digunakan untuk KlikBCA ini akan terputus hubungan network dengan komputer lain secara total. Sebelum menjalankan VPN ini, sebaiknya tutup semua aplikasi jaringan (email client, ftp, dan lainnya) agar tidak terganggu.

Jika login berhasil, akan muncul teks seperti ini:

```
Username []: ibsxxxxxx
Password []:
Authenticating user.
Negotiating security policies.
Securing communication channel.

Welcome to KlikBCA Bisnis Secure
Network

https://ibc.klikbca.com:8002/sme/
login.jsp

Do you wish to continue? (y/n): y

Your VPN connection is secure.

VPN tunnel information.
Client address: 172.16.10.247
Server address: 202.6.211.33
Encryption: 168-bit 3-DES
Authentication: HMAC-MD5
IP Compression: None
NAT passthrough is active on port
TCP 10000
Local LAN Access is disabled
```

Anda bisa melanjutkan dengan me-minimize terminal dan membuka Firefox, lalu masuk di *http://www.klikbca.com/smelogin.html*.

Untuk menutup VPN, buka kembali terminal, dan tekan Ctrl-C. Koneksi VPN akan terputus, dan network kembali normal.

```
Disconnecting the VPN connection.
```

Trik agar eksekusi VPN client tanpa password root

Pada Linux distro Debian dan turunannya (termasuk Ubuntu), yang dapat menjalankan eksekusi sebagai root (dengan menggunakan sudo) adalah user yang dibuat saat instal kali pertama dan user lain yang tercantum di file /etc/sudoers. Sudoers akan ditanya ulang password-nya untuk menjalankan program sebagai root (sudo). User root sendiri di-*disable* secara default saat instalasi.

VPNClient Cisco ini harus dijalankan sebagai root, agar bisa terkoneksi. Agar user biasa (bukan sudoers) bisa ikut menjalankan vpnclient ini, ataupun agar sudoers tidak selalu ditanya password saat akan menjalankannya, kita harus masukkan aplikasi vpnclient ini dalam file sudoers.

Agar dapat eksekusi VPNClient tanpa password root , buka file /etc/sudoers dengan perintah visudo, dan ketik:

[user di ubuntu] ALL= NOPASSWD: /usr/local/bin/vpnclient

Lihat contoh di bawah ini:

```
# /etc/sudoers
#
root ALL=(ALL) ALL
michael ALL= NOPASSWD: /usr/local/
bin/vpnclient
userlain ALL= NOPASSWD: /usr/local/
bin/vpnclient
```

Membuat shortcut di desktop

Agar user mudah untuk terhubung ke Internet banking, buatlah *launcher* (*shortcut*) di desktop Gnome dengan cara sebagai berikut (Jika menggunakan desktop KDE, ikuti petunjuk di KDE):

1. Kecilkan semua window, sehingga hanya desktop yang terlihat.
2. Klik Kanan di tempat kosong, pilih *Create Launcher...*
3. Pilih *Type : Application in Terminal*, ketik nama dengan VPN KlikBCA, command diisi dengan sudo vpnclient klikbca. Jika punya icon khusus untuk ini, silakan tekan tombol no icon. Jika selesai, klik OK.
4. Akan muncul di desktop icon bernama VPN KlikBCA. User bisa langsung pakai dengan klik ini, isi username dan password lalu buka Firefox setelah VPN terhubung.

Bagi yang tidak terbiasa dengan Firefox, bisa menggunakan IEs4Linux. Tutorial instalasinya bisa dibaca di *InfoLINUX* 01/2007.

Dengan suksesnya instalasi ini, Linux Anda sudah bisa digunakan sebagai terminal Internet banking, dalam contoh ini KlikBCA Bisnis.

Indra Sanjaya [indsan@gmail.com]

Mudah Membuat Tutorial dengan Wink

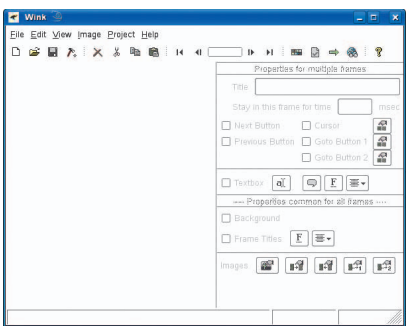
Membuat tutorial secara "live" via teknik *desktop recording* sudah bisa dilakukan dengan mudah di Linux. Dari beberapa cara yang bisa dilakukan, dua di antaranya yang cukup populer adalah menggunakan *software* Wink dan xvidcap. Dalam kesempatan ini, kita akan mencoba menggunakan *free software* Wink sebagai tool desktop recording.

Penulis telah berhasil menggunakan Wink di distro Linux Kubuntu 6.06, SUSE 10.0 dan openSUSE 10.2. Tentu saja Wink bisa diinstal di distro lain secara langsung ataupun dengan modifikasi *library*. Khusus untuk pengguna openSuse 10.2, Wink sudah tidak perlu diinstal secara manual, karena sudah disertakan dalam DVD instalasi (penulis menggunakan DVD dari *InfoLINUX* 02/2007).

Installer Wink bisa didapatkan dari website *www.debugmode.com*. Untuk menginstal secara manual, ekstrak file instalasi dan simpan ke suatu direktori, misal ke folder wink15. Dari console masuk ke folder hasil ekstrak:

```
$ cd wink15
$ ./installer.sh
```

Jika semua dependensi yang dibutuhkan sudah dimiliki, akan muncul pertanyaan folder tujuan instalasi Wink. Gunakan file explorer (Nautilus) dan masuk ke folder hasil instalasi tersebut, kemudian eksekusi file wink. Bagi yang menggunakan OpenSuse 10.2, menu Wink ada di *Applications|Office|Presentation*. Tampilan pertama Wink seperti pada Gambar 1.

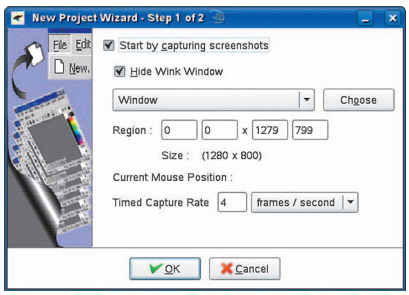


Gambar 1. Tampilan pertama Wink.

Nah, sekarang kita sudah bisa langsung mulai melakukan *desktop recording*. Klik tombol *New Project* atau dari menu *File|New*. Kita akan diberikan New Project Wizard yang sederhana.

Step #1 Proses recording

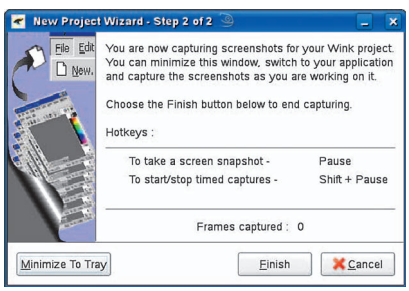
- Pertama-tama kita memilih area yang direkam. Untuk merekam seluruh layar desktop, pilih "Screen". Jika ingin merekam aktivitas di satu software tertentu saja, pilih "Window". Untuk memilih software/window yang ingin direkam



Gambar 2. Wizard membuat project baru.

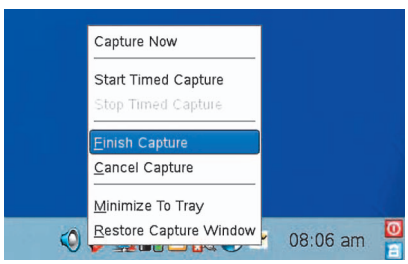
klik "Choose" kemudian klik window yang diinginkan. Pilih "Hide Wink Window" agar window utama Wink hilang dan tidak ikut terrekam. Gunakan Capture Rate = 4 (default) karena hasilnya nanti sudah cukup baik. Setelah itu, klik "OK". Lihat Gambar 2.

- Sebelum mulai merekam, jangan lupa klik "Minimize To Tray". Untuk mulai merekam, tekan "Shift + Pause". Silakan mengoperasikan software yang ingin dibuat tutorialnya. Semua aktifitas visual menggerakan mouse, mengklik tombol, ataupun mengetik keyboard di dalam area *window* yang dipilih akan direkam oleh Wink. Lihat Gambar 3.

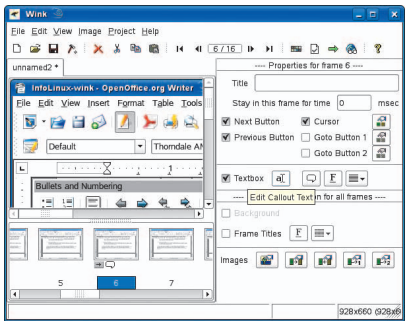


Gambar 3. Saatnya mulai merekam.

- Untuk berhenti merekam, ketik "Shift + Pause" sekali lagi. Kemudian klik kanan pada icon wink di system tray lalu klik "Finish Capture" untuk menyelesaikan wizard recording. Lihat Gambar 4.



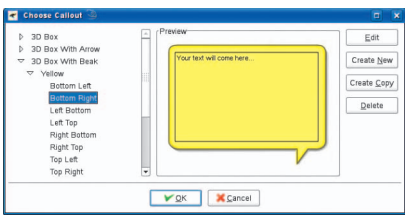
Gambar 4. Mengakhiri rekaman.



Gambar 5. Edit Callout Text.

Step #2 Menambahkan komentar

- Untuk memberikan komentar ke dalam rekaman yang dibuat, klik nomor frame yang ingin diberi komentar, kemudian klik opsi "Textbox". Klik tombol "Edit Callout Text" untuk menulis komentar. Lihat Gambar 5 dan 6.
- Untuk mengubah bentuk *callout*, klik "Choose Callout" dan pilih bentuk callout yang diinginkan. Untuk memberikan komentar di tempat lain, ulangi lagi proses ini pada nomor *frame* yang berbeda. Lihat Gambar 7.

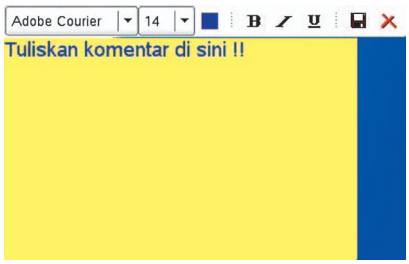


Gambar 7. Mengubah bentu callout.

Step #3 Me-render hasil recording

- Klik "Render". Pilih *output file type* sebagai Macromedia Flash (*.swf). Klik "Browse" untuk menentukan tempat dan nama file. Klik "OK". Biarkan *setting*-an lain secara default. Kemudian klik "OK" lagi untuk melakukan *rendering*. Lihat Gambar 8 dan 9.

Ya, kini kita sudah berhasil menggunakan Wink untuk melakukan desktop recording. Kita bisa menyimpan project ini jika mau, klik *File|Save*. Cukup sederhana, tapi tutorial yang dihasilkan sangat bermanfaat bagi orang-orang yang baru mengenal Linux. Pemanfaatan desktop recording untuk pembuatan tutorial juga sangat baik digunakan untuk mengajarkan cara pengoperasian suatu software secara cepat.



Gambar 6. Kotak edit callaout.

Tentang Output File

Wink bisa menghasilkan tipe file flash (*.swf) maupun executable (*.exe). Secara default, distro seperti openSuse sudah memberikan flash player pada saat instalasi. Hal ini sangat mempermudah dalam pendistribusian tutorial yang dibuat karena sudah banyak Linux yang siap untuk menjalankan flash.

Jika memilih tipe file flash sebagai output, wink secara otomatis membuatkan sebuah file *.htm yang terintegrasi dengan file flash. User tinggal membuka file *.htm tersebut dengan *browser* Firefox atau Mozilla yang sudah dilengkapi dengan *plugin flash*. Keuntungan me-render hasil outputnya sebagai file flash adalah hasil rekaman ini bisa digunakan multiplatform. User cukup memiliki web browser standar yang dilengkapi dengan flash plugin, maka menggunakan sistem operasi manapun user akan bisa melihat hasil rekaman kita.

Adapun untuk Linux yang belum dilengkapi flash player secara default, bisa mendownload file installer flash player langsung dari website pembuat flash player. Cara instalasinya pun sangat mudah. Atau bisa juga melalui cara yang lebih mudah lagi, yaitu dengan melakukan koneksi ke Internet menggunakan browser, misal kita gunakan firefox, dan membuka website yang ada *content* flash-nya. Nanti akan ada permintaan untuk menginstal plugin flash secara otomatis.

Saat ini, kode flash player sudah ada yang diberikan ke pengembang open source (Mozilla). Selain itu, juga sudah ada Gnash yang menjadi alternatif flash player versi open source. Kita juga tidak perlu khawatir jika menyimpan ke dalam file executable. Dengan menginstal wine, tinggal klik pada file executable tersebut, maka secara otomatis wine akan menjalankannya.

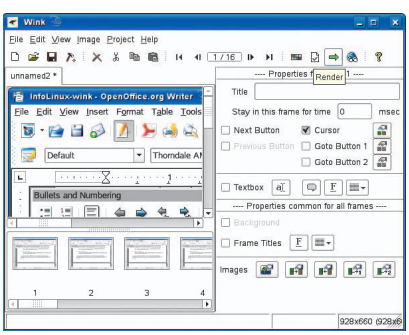
Untuk keperluan belajar bersama, file-file hasil rekaman ini bisa disimpan di server. Misal kita menggunakan web server Apache, file-file hasil rendering tersebut cukup diletakkan pada sebuah folder di document root httpd. User tinggal mengakses folder tersebut, misalnya *http://192.168.1.1/nama_folder* untuk menampilkan seluruh hasil rendering.

Walaupun hasil rendering yang kita peroleh sudah cukup bagus dengan visualisasi yang sangat jelas dan ukuran file cukup kecil, masih terasa ada yang kurang. Alangkah lebih baiknya kalau rekaman yang dihasilkan dilengkapi dengan suara. Misal, jika kita menekan beberapa tombol shortcut keyboard, kita bisa memberikan informasi tersebut melalui audio, karena informasi tombol keyboard mana saja yang kita tekan tidak tampil di layar rekaman.

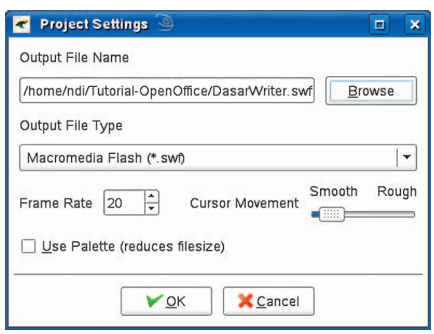
Jika kita membuatnya menggunakan software recording yang hasilnya dalam bentuk file movie, tipe file dan jenis *encoder-decoder*-nya pun perlu diperhatikan. Untuk menghasilkan output rekaman yang berupa movie, kita bisa menggunakan software desktop recording xvidcap atau istanbul.

Jika ingin berpartisipasi pada project pembuatan tutorial open source software, termasuk yang berbasis multimedia seperti Wink ini, silakan bergabung ke milis *opentutorial@yahoogroups.com*.

Yusuf Kurniawan [yusuf132@gmail.com]



Gambar 8. Menyiapkan render.



Gambar 9. Menentukan file output.

Manajemen Kapasitas Harddisk Menggunakan LVM

Bagian 2 dari 2 artikel

Logical Volume Manager atau yang biasa disingkat dengan LVM merupakan metode alokasi kapasitas di suatu media penyimpanan, yang lebih fleksibel daripada skema partisi konvensional. Dengan menggunakan LVM, kita dapat dengan mudah mengubah kapasitas harddisk tanpa perlu khawatir kehilangan data.

Pada edisi sebelumnya, kita telah mengenal bagaimana metode penggunaan dasar LVM. Di edisi ini, kita akan melihat teknik penggunaan metode LVM yang lainnya, seperti cara menambah kapasitas harddisk, LVM dalam RAID, dan sebagainya.

Menambah dan menghapus partisi harddisk

Sejauh ini, kita belum pernah menggunakan partisi /dev/sdf. Untuk itu, sekarang kita akan membuat partisi /dev/sdf1 sebesar 25 GB, dan menambahkannya ke dalam volume group fileserver yang sudah kita buat di edisi sebelumnya.

```
server1:~# fdisk /dev/sdf
.....................................
Command (m for help): n
.....................................
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
```

(Note: ketikkan p, kemudian pilih 1, untuk membuat sebuah partisi di primary partition di partisi ke-1).

```
.....................................
First cylinder (1-10443, default 1):
Using default value 1
Last cylinder or +size or +sizeM or
+sizeK (1-10443, default 10443):
+25000M
```

(Note : Tekan 1 atau isikan 1 untuk first cylinder, kemudian isikan +25000M untuk membuat partisi sebesar 25 GB).

```
.....................................
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 8e
Changed system type of partition 1
to 8e (Linux LVM)
```

(Note: Isikan t untuk memberikan sistem di partisi yang baru saja kita buat. Lalu isikan 8e di option Hex code untuk memberikan Linux LVM sebagai file system-nya).

```
.....................................
Command (m for help): w
The partition table has been
altered!
```

(Note: Isikan w untuk menulis semua perubahan option yang telah dilakukan di fdisk).

```
.....................................
Calling ioctl() to re-read partition
table.
Syncing disks.
```

Jalankan fdisk -l, untuk sekedar memastikan kalau partisi /dev/sdf1 yang baru saja kita buat sudah ada.

```
server1:~# fdisk -l
```

Berikutnya kita akan menyiapkan partisi /dev/sdf1 agar dapat digunakan dalam LVM.

```
server1:~# pvcreate /dev/sdf1
  Physical volume "/dev/sdf1"
successfully created
```

Tambahkan partisi /dev/sdf1 ke dalam volume group fileserver.

```
server1:~# vgextend fileserver
/dev/sdf1
  Volume group "fileserver"
successfully extended
```

Jalankan vgdisplay untuk melihat kapasitas total dari volume group fileserver. Dari hasil output-nya, kita dapat melihat bahwa kapasitas total volume group fileserver telah bertambah daripada kapasitas total sebelumnya.

```
server1:~# vgdisplay
```

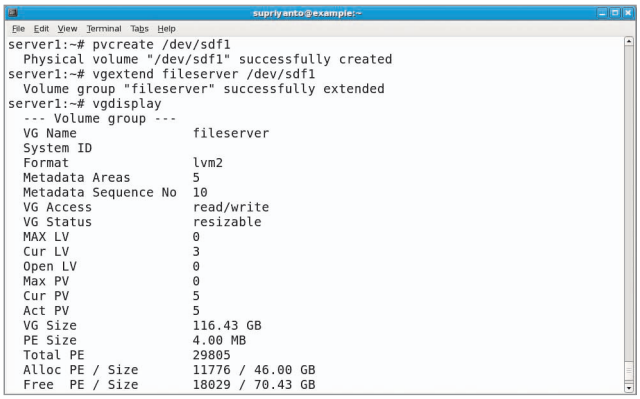
Sekarang kita akan menghapus partisi /dev/sdb1. Tetapi sebelum melakukan hal ini, kita akan meng-*copy* semua data yang terdapat pada partisi tersebut ke partisi /dev/sdf1.

```
server1:~# pvmove /dev/sdb1 /dev/
sdf1
```

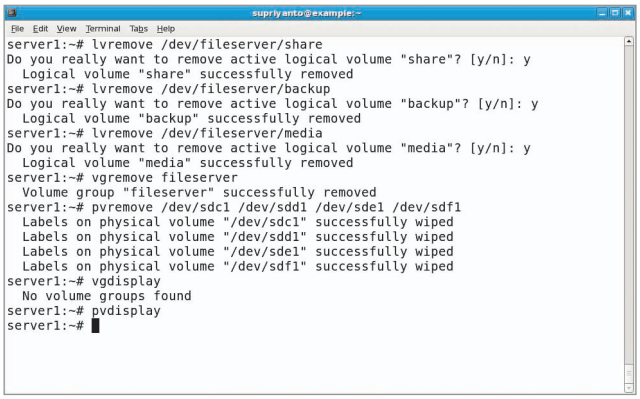
Berikutnya, kita akan menghapus partisi /dev/sdb1 dari volume group fileserver.

```
server1:~# vgreduce fileserver
/dev/sdb1
  Removed "/dev/sdb1" from volume
group "fileserver"
```

Jalankan kembali vgdisplay. Dari hasil *output*-nya, terlihat kalau kapasitas volume group fileserver telah berkurang setelah par-



Menambahkan partisi /dev/sdf ke dalam volume group fileserver.



Menghapus semua logical volume, volume group, dan partisi LVM yang telah dibuat.

tisi /dev/sdb1 dihapus dari volume group fileserver.

```
server1:~# vgdisplay
```

Sekarang kita akan menghapus partisi /dev/sdb1 dari LVM.

```
server1:~# pvremove /dev/sdb1
```

Jalankan perintah pvdisplay untuk melihat apakah partisi /dev/sdb1, sudah tidak terdapat dalam sistem LVM.

```
server1:~# pvdisplay
```

Dari hasil latihan di atas, Anda telah berhasil menghapus partisi /dev/sdb dari sistem LVM. Hal ini membuktikan, kalau proses menambahkan atau menghapus suatu partisi ke dalam sistem LVM, dapat dilakukan secara mudah.

Mengembalikan ke posisi sistem semula

Pada bagian ini, kita akan mengembalikan dahulu sistem yang telah kita buat, ke posisi sistem semula. Langkah ini hanya sebagai latihan, yang bertujuan agar kita dapat mempelajari bagaimana untuk meng-*undo* LVM setup.

Pertama, kita harus melakukan proses unmount terhadap semua logical volume yang sedang dalam proses mount. Jalankan perintah di bawah ini untuk melakukan hal tersebut:

```
server1:~# umount /var/share

server1:~# umount /var/backup

server1:~# umount /var/media
```

Selanjutnya, kita akan menghapus semua logical volume, volume group, dan partisi LVM yang telah dibuat.

```
server1:~# lvremove /dev/fileserver/
share
Do you really want to remove active
logical volume “share”? [y/n]: y
  Logical volume “share”
successfully removed

server1:~# lvremove /dev/fileserver/
backup
Do you really want to remove active
logical volume “backup”? [y/n]: y
  Logical volume “backup”
successfully removed

server1:~# lvremove /dev/fileserver/
media
Do you really want to remove active
logical volume “media”? [y/n]: y
  Logical volume “media”
successfully removed

server1:~# vgremove fileserver
  Volume group “fileserver”
successfully removed

server1:~# pvremove /dev/sdc1 /dev/
sdd1 /dev/sde1 /dev/sdf1
  Labels on physical volume “/dev/
sdc1” successfully wiped
  Labels on physical volume “/dev/
sdd1” successfully wiped
  Labels on physical volume “/dev/
sde1” successfully wiped
  Labels on physical volume “/dev/
sdf1” successfully wiped

server1:~# vgdisplay
  No volume groups found

server1:~# pvdisplay
```

Berikutnya, kita akan mengembalikan isi file /etc/fstab, ke isi awal file /etc/fstab semula, untuk melindungi agar system tidak mencoba untuk melakukan proses mount devices yang sudah tidak ada. Untuk itu, berikan tanda remark (#), pada bagian berikut di dalam file /etc/fstab :

```
server1:/etc# vim /etc/fstab
.....................................
#/dev/fileserver/share   /var/share
ext3       rw,noatime    0 0
#/dev/fileserver/backup  /var/backup
xfs        rw,noatime    0 0
#/dev/fileserver/media   /var/media
reiserfs   rw,noatime    0 0
```

Save dan lakukan proses *reboot* sistem.

```
server1:~# shutdown -r now
```

Setelah masuk ke dalam sistem kembali, coba ketikkan perintah df -h untuk memastikan kalau sistem hanya melakukan mount seperti kondisi awal.

```
server1:~# df -h
```

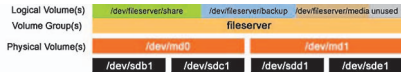
Saat ini, kondisi sistem sudah kembali ke posisi semula. Hanya saja partisi /dev/sdb1-/dev/sdf1 tetap ada, dan direktori /var/share, /var/backup, dan /var/media, tidak perlu kita delete lebih dahulu.

LVM di RAID1

Pada bagian ini, kita akan mempelajari cara konfigurasi LVM dan memindahkannya ke RAID1 array agar terjamin ketersediaannya. Sebagai gambaran, sistem yang kita buat nantinya akan terlihat seperti gambar Skema LVM Full RAID1.

Dari gambaran tersebut, kita akan membuat RAID array /dev/md0 dari partisi /dev/sdb1+/dev/sdc1, dan RAID array /dev/

md1 dari partisi /dev/sdd1+/dev/sde1. Partisi /dev/md0 dan /dev/md1, nantinya akan menjadi physical volume untuk LVM.



Skema LVM Full RAID1.

Sebelum melakukan hal tersebut, kita akan mengonfigurasi LVM sebagai berikut:

```
server1:~# pvcreate /dev/sdb1 /dev/
sdc1 /dev/sdd1 /dev/sde1

server1:~# vgcreate fileserver /dev/
sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1

server1:~# lvcreate --name share --
size 40G fileserver

server1:~# lvcreate --name backup
--size 5G fileserver

server1:~# lvcreate --name media --
size 1G fileserver

server1:~# mkfs.ext3 /dev/
fileserver/share

server1:~# mkfs.xfs /dev/fileserver/
backup

server1:~# mkfs.reiserfs /dev/
fileserver/media

Lakukan proses mount kembali logical
volume yang telah dibuat.
server1:~# mount /dev/fileserver/
share /var/share

server1:~# mount /dev/fileserver/
backup /var/backup

server1:~# mount /dev/fileserver/
media /var/media
```

Sekarang kita akan memindahkan seluruh isi dari partisi /dev/sdc1 dan /dev/sde1 (/dev/sdc1 merupakan partisi kedua dari RAID array /dev/md0, sedangkan /dev/sde1 merupakan partisi kedua dari RAID array /dev/md1) ke partisi sisa-nya. Karena setelah ini kita akan menghapus partisi tersebut dari LVM, dan memformatnya dengan tipe fd (Linux RAID autodetect), dan memindahkan dari /dev/md0 yang direpresentasikan ke /dev/md1. Untuk melakukan hal ini, jalankan perintah berikut:

```
server1:~# modprobe dm-mirror

server1:~# pvmove /dev/sdc1

server1:~# vgreduce fileserver
/dev/sdc1

server1:~# pvremove /dev/sdc1

server1:~# pvdisplay

server1:~# pvmove /dev/sde1

server1:~# vgreduce fileserver
/dev/sde1

server1:~# pvremove /dev/sde1

server1:~# pvdisplay
```

Berikutnya, kita akan memformat partisi /dev/sdc1 dengan menggunakan tipe fd (Linux RAID autodetect).

```
server1:~# fdisk /dev/sdc
.....................................
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1
to fd (Linux raid autodetect)
Command (m for help): w
The partition table has been
altered!

Calling ioctl() to re-read partition
table.
Syncing disks.
```

Berikutnya kita akan melakukan hal yang sama terhadap partisi /dev/sde1.

```
server1:~# fdisk /dev/sde
```

Saat Anda mengetikkan fdisk -l, Anda dapat melihat bahwa partisi /dev/sdc1 dan partisi /dev/sde1 sudah berubah menjadi file system Linux RAID autodetect.

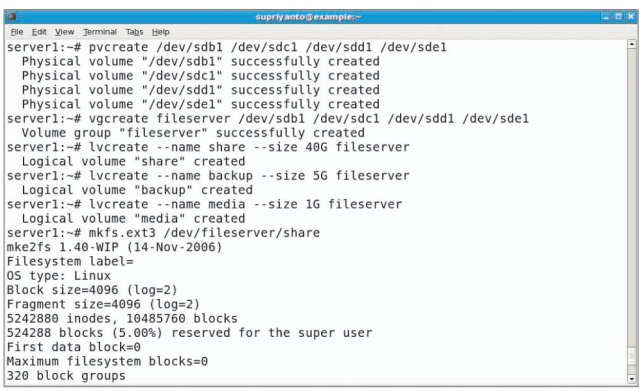
```
server1:~# fdisk -l
```

Sekarang kita akan menambahkan /dev/sdc1 ke /dev/md0 dan /dev/sde1 ke /dev/md1. Karena node kedua (/dev/sdb1 dan /dev/sdd1) belum siap, maka kita harus menjelaskannya dengan perintah berikut:

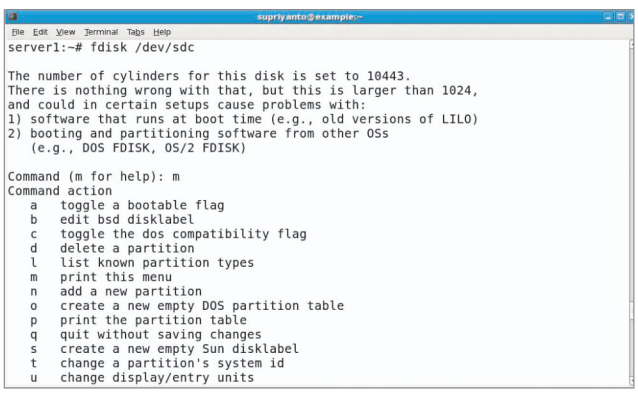
```
server1:~# mdadm --create /dev/md0
--auto=yes -l 1 -n 2 /dev/sdc1
missing
mdadm: array /dev/md0 started.

server1:~# mdadm --create /dev/md1
--auto=yes -l 1 -n 2 /dev/sde1
missing
mdadm: array /dev/md1 started.
```

Next, kita akan menyiapkan /dev/md0 dan /dev/md1 untuk partisi LVM.

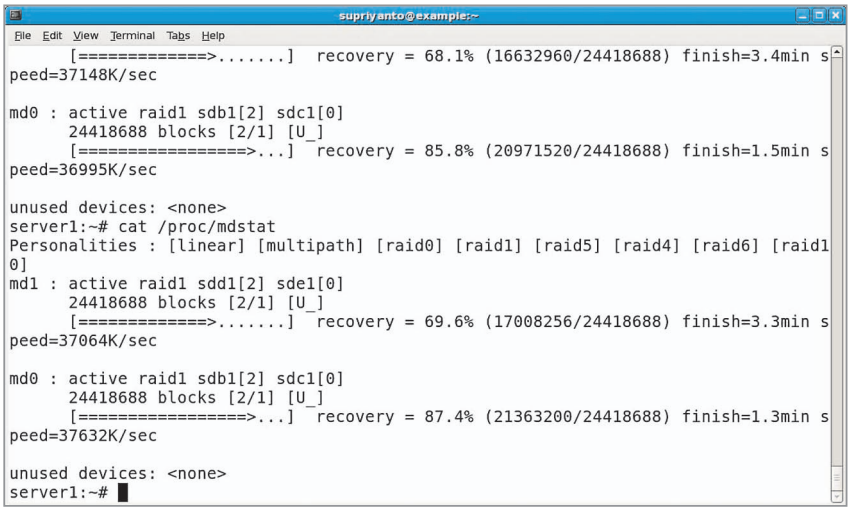
```
server1:~# pvcreate /dev/md0 /dev/
```



Proses konfigurasi LVM kembali.



Mengubah file system /dev/sdc menjadi Linux RAID autodetect.



Proses sinkronisasi RAID array.

```
md1

server1:~# vgextend fileserver
/dev/md0 /dev/md1

server1:~# pvdisplay

server1:~# vgdisplay
```

Sekarang kita akan memindahkan isi dari partisi /dev/sdb1 ke /dev/md0 dan isi dari /dev/sdd1 ke /dev/md1, kemudian hapus partisi /dev/sdb1 dan /dev/sdd1 dari LVM.

```
server1:~# pvmove /dev/sdb1 /dev/md0

server1:~# pvmove /dev/sdd1 /dev/md1

server1:~# vgreduce fileserver
/dev/sdb1 /dev/sdd1

server1:~# pvremove /dev/sdb1 /dev/
sdd1
```

Sekarang hanya /dev/md0 dan /dev/md1 yang terdapat di physical volumes.

```
server1:~# pvdisplay
```

Lanjutkan dengan melakukan format partisi /dev/sdb1 dan sdd1 dengan tipe fd (Linux RAID autodetect).

```
server1:~# fdisk /dev/sdb
.................................
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1
to fd (Linux raid autodetect)
.................................
Command (m for help): w
The partition table has been
altered!

Calling ioctl() to re-read partition
table.
Syncing disks.
```

Lakukan hal yang sama terhadap /dev/sdd.

```
server1:~# fdisk /dev/sdd
```

Tambahkan /dev/sdb1 ke /dev/md0 dan /dev/sdd1 ke /dev/md1.

```
server1:~# mdadm --manage /dev/md0
--add /dev/sdb1
mdadm: added /dev/sdb1

server1:~# mdadm --manage /dev/md1
--add /dev/sdd1
```

Sekarang kedua RAID array telah di sinkronisasi. Langkah ini akan memakan sedikit waktu, dan Anda dapat mengeceknya dengan menjalankan perintah berikut:

```
server1:~# cat /proc/mdstat
.................................
md1 : active raid1 sdd1[2] sde1[0]
      24418688 blocks [2/1] [U_]
      [==================>.]
recovery = 99.5% (24304768/24418688)
finish=0.0min speed=74082K/sec
.................................
unused devices: <none>
```

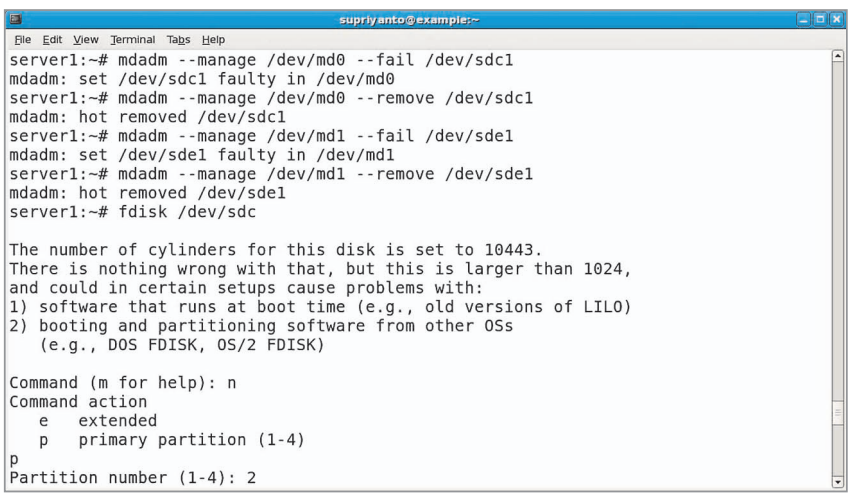
Dan akan terlihat seperti berikut jika proses sinkronisasi RAID array telah selesai.

```
server1:~# cat /proc/mdstat
Personalities : [linear] [multipath]
[raid0] [raid1] [raid5] [raid4]
[raid6] [raid10]
md1 : active raid1 sdd1[1] sde1[0]
      24418688 blocks [2/2] [UU]

md0 : active raid1 sdb1[1] sdc1[0]
      24418688 blocks [2/2] [UU]

unused devices: <none>
```

Jalankan perintah pvdisplay untuk melihat kapasitas total dari Physical Volume.

```
server1:~# pvdisplay
```

Dari hasil output pvdisplay, dapat terlihat kalau ukuran total physical volume tersedia sebesar 2 * 23.29GB = 46.58GB. Jadi kita masih memiliki space kosong sebesar 0.5GB, yang dapat diberikan ke salah satu logical volume yang ada (40 GB (share),



Menandai dan menghapus partisi /dev/sdc1 dan /dev/sde1 ke dalam RAID array.

5GB (backup), 1GB (media) = 46 GB). Sebagai contoh di sini, kita akan memberikan kapasitas 0.5GB pada partisi media.

```
server1:~# lvextend -L1.5GB /dev/
fileserver/media
```

```
server1:~# resize_reiserfs /dev/
fileserver/media
```

Jika Anda menginginkan agar logical volume dapat di-*mount* secara otomatis setiap kali booting, maka Anda harus memodifikasi isi file /etc/fstab kembali (dengan jalan menghilangkan kembali tanda remark (#), pada bahasan sebelumnya).

Setelah diedit, lakukan reboot system

```
server1:~# shutdown -r now
```

Mengganti kapasitas harddisk

Saat ini kita sudah menggunakan empat buat harddisk dengan kapasitas masing-masing harddisk sebesar 25 GB. Sekarang kita asumsikan kapasitas harddisk sudah tidak mencukupi lagi, dan kita membutuhkan lebih banyak kapasitas untuk konfigurasi RAID ini. Maka dari itu, kita akan mengganti kapasitas harddisk 25 GB yang kita miliki dengan harddisk 80 GB. Dengan cara ini, sekarang kita tetap dapat menggunakan kapasitas harddisk yang sudah ada, ditambah dengan kapasitas harddisk yang baru ditambah.

Prosedur yang akan kita lakukan adalah sebagai berikut: pertama remove /dev/sdb dan /dev/sdd dari RAID array, ganti dengan harddisk yang lebih besar, kemudian letakkan kembali ke dalam RAID array. Hal ini yang akan kita lakukan juga untuk /dev/sdc dan /dev/sde.

Untuk melakukan hal ini, lakukan langkah berikut. Pertama, kita akan menandai kalau /dev/sdb1 sebagai failed.

```
server1:~# mdadm --manage /dev/md0
--fail /dev/sdb1
```

Maka saat menjalankan perintah cat /proc/mdstat, hasil output-nya akan terlihat sebagai berikut:

```
server1:~# cat /proc/mdstat
Personalities : [linear] [multipath]
[raid0] [raid1] [raid5] [raid4]
[raid6] [raid10]
md0 : active raid1 sdc1[0]
sdb1[2](F)
      24418688 blocks [2/1] [U_]

md1 : active raid1 sde1[0] sdd1[1]
      24418688 blocks [2/2] [UU]

unused devices: <none>
```

Selanjutnya kita akan menghapus /dev/sdb1 dari RAID array /dev/md0.

```
server1:~# mdadm --manage /dev/md0
--remove /dev/sdb1
```

Maka saat menjalankan perintah cat /proc/mdstat, hasil outputnya akan terlihat sebagai berikut:

```
server1:~# cat /proc/mdstat
Personalities : [linear] [multipath]
[raid0] [raid1] [raid5] [raid4]
[raid6] [raid10]
md0 : active raid1 sdc1[0]
      24418688 blocks [2/1] [U_]

md1 : active raid1 sde1[0] sdd1[1]
      24418688 blocks [2/2] [UU]

unused devices: <none>
```

Lakukan hal yang sama untuk /dev/sdd1.

```
server1:~# mdadm --manage /dev/md1
--fail /dev/sdd1

server1:~# mdadm --manage /dev/md1
--remove /dev/sdd1
```

Dalam sistem sebenarnya, mungkin kita harus melakukan *shutdown* komputer, mengganti kapasitas harddisk 25 GB di /dev/sdb dan /dev/sdd dan menggantinya dengan kapasitas 80GB. Namun di sistem LVM+RAID ini, kita tidak perlu melakukan hal ini karena semua harddisk saat ini telah memiliki kapasitas 80 GB.

Selanjutnya kita harus melakukan format /dev/sdb dan /dev/sdd. Kita harus membuat partisi /dev/sdb1 resp. ke /dev/sdd1, menggunakan tipe fd (Linux RAID autodetect), kapasitas 25 GB (setting yang sama seperti di harddisk yang lama), dan /dev/sdb2 resp. partisi /dev/sdd2. type fd, dan mencangkup diharddisk. Sama halnya dengan /dev/sdb1 dan /dev/sdd1 yang ditampilkan dalam harddisk. kita juga perlu menciptakan /dev/sdb2 dan /dev/sdd2 di special example.

```
server1:~# fdisk /dev/sdb
.................................
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (3041-10443, default
3041):
Using default value 3041
Last cylinder or +size or +sizeM or
+sizeK (3041-10443, default 10443):
Using default value 10443

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): fd
Changed system type of partition 2
to fd (Linux raid autodetect)

Command (m for help): w
The partition table has been
altered!

Calling ioctl() to re-read partition
table.
Syncing disks.
```

Lakukan hal yang sama untuk /dev/sdd.

```
server1:~# fdisk /dev/sdd
```

Berikutnya, kita akan menambahkan /dev/sdb1 ke /dev/md0 kembali, dan /dev/sdd1 ke /dev/md1.

```
server1:~# mdadm --manage /dev/md0
--add /dev/sdb1
mdadm: re-added /dev/sdb1

server1:~# mdadm --manage /dev/md1
--add /dev/sdd1
mdadm: re-added /dev/sdd1
```

Sekarang isi antara RAID array akan disinkronisasikan. Tunggu sampai proses sinkronisasi RAID array selesai. Kita dapat mengecek status sinkronisasi ini dengan menggunakan perintah berikut:

```
server1:~# cat /proc/mdstat
Personalities : [linear] [multipath]
[raid0] [raid1] [raid5] [raid4]
[raid6] [raid10]
md0 : active raid1 sdb1[1] sdc1[0]
      24418688 blocks [2/2] [UU]

md1 : active raid1 sdd1[1] sde1[0]
      24418688 blocks [2/2] [UU]
```

```
unused devices: <none>
```

Sekarang kita akan melakukan proses yang sama kembali. Kali ini untuk mengganti /dev/sdc dan /dev/sde. Lakukan perintah-perintah di bawah ini:

```
server1:~# mdadm --manage /dev/md0
--fail /dev/sdc1

server1:~# mdadm --manage /dev/md0
--remove /dev/sdc1

server1:~# mdadm --manage /dev/md1
--fail /dev/sde1

server1:~# mdadm --manage /dev/md1
--remove /dev/sde1

server1:~# fdisk /dev/sdc
```

(Note: Berikan kapasitas sisa sebagai partisi /dev/sdc2, dengan tipe file sistem fd.)

```
server1:~# fdisk /dev/sde
```

(Note : Berikan kapasitas sisa sebagai partisi /dev/sde2, dengan tipe file sistem fd.)

Lihat proses sinkronisasi dengan menggunakan cat /proc/mdstat, dan tunggu sampai proses sinkronisasi selesai.

```
server1:~# cat /proc/mdstat
```

Berikutnya, kita akan membuat RAID array /dev/md2 dari /dev/sdb2 dan /dev/sdc2, dan RAID array /dev/md3 dari /dev/sdd2 dan /dev/sde2.

```
server1:~# mdadm --create /dev/md2
--auto=yes -l 1 -n 2 /dev/sdb2
/dev/sdc2
mdadm: array /dev/md2 started.

server1:~# mdadm --create /dev/md3
--auto=yes -l 1 -n 2 /dev/sdd2
/dev/sde2
mdadm: array /dev/md3 started.
```

RAID array yang baru, seharusnya sudah dapat disinkronisasikan sekarang. Silakan di check dengan menggunakan perintah:

```
server1:~# cat /proc/mdstat
```

Setelah proses sinkronisasi selesai, kita akan menyiapkan /dev/md2 dan /dev/md3 untuk LVM.

```
server1:~# pvcreate /dev/md2 /dev/
md3
  Physical volume "/dev/md2"
successfully created
  Physical volume "/dev/md3"
successfully created
```

Selanjutnya, tambahkan /dev/md2 dan /dev/md3 ke dalam volume group fileserver.

```
server1:~# vgextend fileserver
/dev/md2 /dev/md3
  Volume group "fileserver"
successfully extended
```

Kita dapat melihat dengan menggunakan perintah pvdisplay, bahwa phsyical volume /dev/md2 dan /dev/md3, telah masuk ke dalam volume group fileserver.

```
server1:~# pvdisplay
```

Sekarang, kita telah sukses dalam melakukan penggantian kapasitas harddisk berukuran kecil dengan yang lebih besar. Kita juga telah memiliki kapasitas harddisk yang lebih besar (2*23.29GB+2*56.71GB=160GB). Dengan ini, kita dapat memberikan kapasitas tambahan kepada logical volume yang ada. Sebagai contoh, kita akan menambahkan kapasitas logical volume backup.

```
server1:~# lvextend -L10G /dev/
fileserver/backup
  Extending logical volume backup
to 10.00 GB
  Logical volume backup successfully
resized
```

Untuk menambah kapasitas xfs filesystem, jalankan perintah berikut:

```
server1:~# xfs_growfs /dev/
fileserver/backup
```

Dengan menggunakan perintah df -h, Anda dapat melihat bahwa kapasitas logical volume back-up telah bertambah menjadi 10 GB.

```
server1:~# df -h
```

Jika Anda telah melalui semua proses ini dengan benar, berarti Anda telah berhasil mengonfigurasi LVM dan LVM dalam RAID. Dengan memahami cara penggunaan dan konfigurasi LVM dan LVM dalam RAID di Linux, diharapkan kita dapat menangani permasalahan kapasitas harddisk secara mudah. Aplikasi LVM dan RAID di Linux sudah cukup mapan, sehingga cukup baik diterapkan untuk kebutuhan perusahaan. Akhir kata, selamat mencoba!

Supriyanto [supriyanto@infolinux.co.id]