

Open Source Programming dengan bahasa C/C++

Muchammad Aiman

islamthink@gmail.com

http://islamthink.wordpress.com

Lisensi Dokumen:

Copyright © 2003-2007 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Kali ini kita akan membahas tentang open-source programming menggunakan bahasa pemrograman C. Ini tentunya karena Linux yang open-source sudah banyak menggunakan C untuk membangun sistemnya. Tentunya jika kita menggunakan C dalam program open kita, maka itu akan disupport penuh oleh Linux (*nix) dan team developernya. Oks.

Index

1. Overview
2. Membangun Open-Source Program
3. Build and run
4. Membuat Packet
5. Penutup
6. Referensi

1. Overview

Kali ini kita akan masuk ke pembahasan yang rupanya agak penting bagi kita komunitas Linux-er dan sistem administrator. Hal ini tentunya karena kadang kita membutuhkan program untuk membantu pekerjaan sehari-hari kita sbg sistem administrator atau sekedar untuk coba-coba membuat tools kecil untuk pentest. Yah, tentunya menggunakan bahasa C karena 'its powerful'.

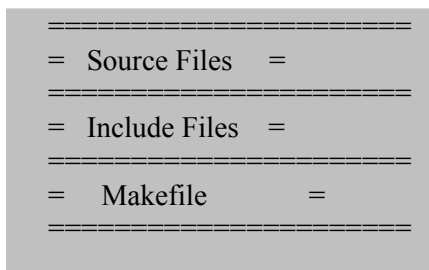
C di Linux bagai keju dalam roti tawar, wew :o Sangat support satu dengan yang lain. Lihat saja source linux di /usr/src/... . Semuanya dibangun dengan bahasa pemrograman C. Nah, tentunya kalau kita sudah memahami cara kerja C di Linux, kita juga bisa menggunakan source-source itu untuk kepentingan kita

sendiri. Menggunakan source orang lain untuk memperkaya program buatan kita. Bayangkan saja, banyaknya program yang dibuat untuk jalan di Linux, seperti Nmap, OpenSSL, binutils, XMMS, MPlayer, dan koleksi program-program yang lain. Semuanya itu dibuat dengan C. Dengan sources yang sudah ada itu, kita benar-benar bisa membuat program baru yang lebih unik dan penuh fungsionalitas tergantung keinginan kita.

Misalnya Clamav, salah satu antivirus yang jalan di Linux-like OS. Di dalamnya ada banyak sekali library dan sources yang bisa kita gunakan ulang pada program-program kita. Seperti source untuk compressing file/direktori dan ekstraksi. Sources untuk meng-inspeksi berbagai tipe file seperti PDF, EXE, CHM, HTML, dan lainnya. Tentunya semua itu bisa dengan lebih mudah kita lakukan jika kita paham konsep dasar OpenSource Programming, sekarang dengan Linux.

2. Elemen Open-Source Program

Perlu sekali kita mengetahui lingkungan dan elemen-elemen yang akan sering sekali kita temui pada programming open-source. Elemen-elemen di sini adalah poin-poin seperti Source files, Include files, dan Makefile. Ketiganya memiliki manajemen sendiri yang saling berhubungan sehingga program C kita terbentuk.



2.1 Source Files dan Include Files

Source program adalah file-file sumber dimana kita menuliskan program kita di dalamnya. Perlu diketahui, Source File (SF) tidak harus ditulis semua pada satu file. Nah, inilah yang disebut me-manage source file. Dengan memanage inilah kita bisa membuat program yang layak disebut sebagai opensource. Karena orang lain yang membaca sourcenya dapat dengan mudah memahami jalannya program, dan memodifikasinya. Untuk contoh arsitektur source file yang lumayan baik, lihat contoh di bawah.

```
/* Program Virus Scanner */
VScan/
-----
    include/
        vscan-conf.h

        tipefile.c
        tipefile.h

        options.c
        options.h

        readdb.c
        readdb.h
```

```
scan.c
scan.h
-----
vscan.c
Makefile
```

Sekilas melihat arsitektur di atas, terlihat sederhana. Menyederhanakan masalah akan mempermudah menyusun logika. Begitu katanya. Pada susunan di atas, kita mempunyai file source utama (main) yaitu vscan.c. Kemudian, vscan.c ini sangat tergantung (terikat) dengan file-file lain yang dibuat pada direktori include. Fungsi-fungsi yang dijalankan oleh vscan.c telah dideklarasikan dan didefinisikan pada file-file yang ada pada direktori include, baik file *.c maupun file *.h.

Apa yang perlu diperhatikan dengan file *.c dan *.h di dalam include directory. Lihat schema di bawah :

```
include_file.h ----> menuliskan hanya prototipe-nya saja
                   ----> prototipe = <tipe> nama_fungsi(arguments);
                   ----> (=deklarasi)
include_file.c ----> menulis kembali prototipe fungsi dan
                   kode lengkap pembentuknya
                   ----> (=definisi)
```

Uah, belum ngantuk kan! File *.h hanya berisi deklarasi fungsi dan macro define yang dibutuhkan. Lihat contoh di bawah untuk file scan.h dan scan.c .

```
/* File scan.h akan berisi macro define dan nama_fungsi(argumen) saja */

#include "vscan-conf.h"

#define RECURSIVE_DIRECTORY 100
#define FILETYPE_ONE 0x1
#define FILETYPE_TWO 0x2
#define FILETYPE_THREE 0x3
#define FILETYPE_GABUNGAN 0x1 | 0x2 | 0x3

static int scandesc(int deskriptor, const char *opsi);
int scanfile(const char *namafile, const char *opsi);
int scandir(const char *namadir, int recursive, const char *opsi);

/* Selesai sudah untuk file scan.h */

/* File scan.c akan berisi program lengkap dari fungsi-fungsi yang */
/* sudah dideklarasikan sebelumnya, yaitu pada scan.h */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <dirent.h>

#include "tipefile.h"
```

```
#include "options.h"

// masuk ke fungsi-fungsinya ya...
static int scandesc(int deskriptor, const char *opsi)
{
    /* file ini akan men-scan deskriptor dari
    file yang sudah dibuka sebelumnya
    Deskriptor biasanya bertipe integer, biasa
    dikasih nama desc, fd, sockfd, dll */

    int i=0
    while (!eof(deskriptor)) {
        cari_virus(deskriptor);
    }

    /* Contoh scandesc selesai */
}

int scanfile(const char *namafile, const char *opsi)
{
    int desc;

    desc = open(namafile, O_RDONLY);
    if(scandesc(desc, FILETYPE_GABUNGAN)) {
        printf("Ada virus pada %s\n", namafile);
        return (-3);          //kenapa -3 ? bebas, bung!
    }

    return 0;
}

int scandir(const char *namadir, int recursive, const char *opsi)
{
    struct dirent *dir;
    dir = opendir("/etc");

    while(readdir(dir)) {
        if( scanfile(dir.f_name, opsi) == -3) {
            printf("Direktori ga aman\n");
            return (-5);
        }
    }

    return 0;
}

/* Nah, selesai untuk scan.c */
```

Lihat pada dua file scan.h dan scan.c di atas. Bagaimana hubungan antara keduanya. Coba dilihat-lihat dulu beberapa kali. Maka anda bisa mengetahui apa bedanya sekarang. File .c berisi fungsi lengkap yang kita inginkan untuk melakukan tugas-tugas yang diinginkan.

Perlu diketahui, sebenarnya menulis program tanpa membuat file .h boleh-boleh saja. Dan program pun masih bisa di-compile dan di-run. Tetapi kayaknya kurang asyik dan menyalahi standar open-source C programming ;p Masukkan saja semuanya ke file .c kemudian compile, kompilasi pun selesai tanpa error. Tapi file ini akan jadi sangat panjang dan susah untuk didebug. Kecuali memang program kita sangat sangat sederhana atau kecil!

Apa isi main file .c , yang mana di atas kita beri nama vscan.c ? File vscan.c adalah file utama, program yang akan kita jalankan setelah selesai di-compile. Kita akan mengcompile file vscan.c dan menghasilkan file program executable vscan. Secara manual ini bisa dilakukan dengan menjalankan perintah seperti di bawah :

```
gcc -o vscan vscan.c -Iinclude/
```

Kurang lebih begitu. Coba saja, pasti error! Karena memang source kita di atas tidak lengkap dan hanya contoh saja. Tapi jika program kita lengkap, baris perintah di atas Insya Allah berjalan normal-normal saja. Tambahan -I<direktori> akan memberi informasi ke compiler gcc untuk mencari file-file include pada direktori include/ . Sedangkan -o akan memberikan keluaran program setelah dikompile, yaitu vscan. Bebas kok, vscan bisa diganti dengan apapun, yang harus benar adalah file sourcenya, yaitu vscan.c

2.2 Makefile

Makefile adalah file bantuan untuk melakukan instalasi program kita. Instalasi untuk source packet kita terdiri dari :

- Compile : meng-compile file source
- Copy (install) : meng-copy file program ke direktori path Linux (/usr/local/bin)
- Remove (uninstall) : meng-uninstall / menghapus file program dari direktori path
- Clean : membersihkan file-file object dari source direktori
- All (optional) : melakukan compile, copy, dan clean pada satu perintah make

Nah, kita akan membahas elemen-elemen yang berhubungan dengan Makefile. Makefile sudah digunakan oleh Unix dan Linux sejak lama untuk membantu instalasi program pada sistemnya. Karena program buatan kita adalah berbasis source , bukan binary, maka harus ada tahapan bernama Compile atau kompilasi. Tahap ini sama dengan baris perintah yang sudah kita tulis di atas :

```
gcc -o nama_program nama_program.c -Iinclude
```

Tidak harus seperti di atas, tetapi sepertinya itu bentuk paling sederhana. Bisa saja ditambah opsi seperti untuk load library dengan opsi -l<nama_library>. Baris kompilasi ini tentunya akan

direktori utama VScan dan include juga Makefile-nya, maka program open-source kita sudah selesai. Untuk melakukan instalasi seperti di bawah:

```
# make
# make install
# (tidak harus!) make clean

atau -----

# make all
```

Jika kita akan membuat paket program kita, gunakan tar dan bzip2 untuk mengkompres menjadi satu file saja. Jika dibutuhkan, tinggal diekstrak lagi oleh pengguna.

4. Membuat Packet

Setelah kita selesai membuat source dan Makefile dan mencoba menginstallnya pada sistem kita sendiri, maka sekarang saatnya membuatnya satu paket sendiri. Tujuannya tentu agar bisa diupload dan digunakan oleh siapapun yang membutuhkannya. Ikuti baris-baris berikut :

```
(Jika anda sedang berada di direktori VScan/
# cd .. (ini untuk keluar dari direktori utama)
# ls
  VScan

# tar cvf vscan-0.1.tar VScan
# bzip2 vscan-0.1.tar
# ls
  VScan vscan-0.1.tar.bz2
```

Nah, kita sudah memiliki paket lengkap program kita disimpan pada file bernama **vscan-0.1.tar.bz2**.

5. Penutup

Ternyata membuat program open-source tidak sesulit yang kita bayangkan. Yang perlu diperhatikan adalah pertama, logika atau konsep membuat program yang akan kita terapkan selama membuat program. Ditambah banyak latihan, maka insya Allah kita akan lebih mudah dan terbiasa membuat tool dan program-program pada sistem Linux dan Linux-like buatan kita sendiri. Insya Allah pada kesempatan lain kita akan belajar membuat program open-source secara live, dengan menggunakan library-library bawaan dan library milik tools lain pada program kita.

6. Referensi

Schildt, Herbert. THE ANNOTATED ANSI C STANDARD. McGraw-Hill. 1990
Gkioulekas, Eleftherios. Developing Software with GNU,.University of Washington. 1999

Biografi Penulis



Muchammad Aiman. Penulis adalah seorang mahasiswa tingkat akhir pada Perguruan Tinggi Institut Teknologi Telkom Jurusan Teknik Informatika. Tengah menyusun tugas akhir tentang keamanan komputer dengan judul Perancangan dan Implementasi Antivirus Update dengan Universal Packet Update. Di sela-sela kesibukan mengerjakan Tugas Akhir masih berusaha menyempatkan menulis artikel-artikel dalam bidang sistem operasi dan keamanan sistem komputer. *Artikel-artikel yang lainnya dapat didapat gratis di <http://islamthink.wordpress.com>.*