

# Pembuatan Pustaka Turbo / Borland C Untuk GCC Dengan Menerapkan Konsep Pemrograman Modul Banyak

Victor Nommmeinota Papilaya  
victor\_papilaya@yahoo.com

## **Lisensi Dokumen:**

Copyright © 2006 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**.

## **Intisari**

Makalah ini membahas bagaimana membuat program bahasa C menggunakan gcc namun *source* yang dihasilkan dapat juga dijalankan menggunakan Turbo C ataupun Borland C. Solusi yang ditawarkan adalah membuat pustaka baru (dengan memanfaatkan perintah-perintah di gcc) yang berisi perintah-perintah standar di Turbo / Borland C, menggunakan teknik pemrograman modul banyak.

**Kata kunci** : pemrograman modul, Pustaka Turbo C

## **1. Pendahuluan**

IGOS – *Indonesia Go Open Source* telah dikumandangkan. Namun Dominasi dari Microsoft, masih membayangi langkah dunia pendidikan. Masih banyak tugas yang diberikan oleh pengajar, mengharuskan mahasiswa untuk tetap menggunakan program komersial yang berjalan di atas sistem operasi windows, padahal sebenarnya tugas tersebut masih bisa diselesaikan menggunakan program yang non-komersial, jika pengajar mau memberikan sedikit kelonggaran bagi mahasiswa untuk mengeksplorasi dunia *Open Source*.

Bahasa C, saat ini menjadi bahasa pengantar bagi pemula untuk memasuki dunia pemrograman. Untuk mempelajari bahasa ini, biasanya digunakan IDE (*integrated Development Environment*) seperti Turbo C, borland C atau visual C yang berjalan di atas sistem operasi windows. Terdapat alternatif lain yang dapat ditempuh untuk mempelajari bahasa C dan sekaligus sebagai wujud nyata dukungan dunia pendidikan terhadap proses migrasi ke *Open Source*, yaitu dengan mempelajari Bahasa C menggunakan gcc (*GNU C Compiler*) yang berjalan di atas sistem operasi linux dan merupakan *portable compiler* untuk C, C++ dan *Objective C* [1].

Apikasi dasar Bahasa C yang dibuat dengan gcc, bisa dipastikan dapat berjalan dengan baik pada *compiler* C yang lain, karena adanya standarisasi. Namun jika pemrogram sudah mulai menggunakan fungsi-fungsi tertentu (bukan fungsi standar), seperti gotoxy yang dikenal dalam IDE Turbo C, maka *source* tersebut tidak akan bisa dijalankan menggunakan gcc. Agar *source* tersebut bisa dijalankan di gcc,

maka harus dibuat pustaka baru yang berisi fungsi gotoxy. Pada tulisan ini akan dibahas proses pembuatan pustaka baru yang berisi perintah/fungsi yang terdapat di Turbo / Borland C menggunakan teknik pemrogram modul banyak.

## 2. Pemrogram Modul Dalam C [2]

### 2.1 Pemrograman Modul Tunggal

Berikut ini adalah sebuah contoh program C yang berfungsi membalikan sebuah string. Program ini dibuat dalam mode *Single Module*, yang artinya baik fungsi maupun main program, diletakan di dalam *file* yang sama.

```
#include <stdio.h>
void reverse(char *before, char *after);
int main() {
    char str[100];
    reverse("cat",str);
    printf("reverse (\"cat\") = %s\n",str);
    reverse("noon",str);
    printf("reverse(\"noon\") = %s\n", str);
    return 0;
}
void reverse (char *before, char *after) {
    int i;
    int j;
    int len;

    len=strlen(before);

    for(j=len-1,i=0;j>=0;j--,i++)
        after[i]=before[j];

    after[len] = NULL
}
```

### 2.2 Pemrogram Modul Banyak

Program yang telah dibuat sebelumnya sama sekali tidak ada yang salah. Namun akan sulit untuk menggunakan fungsi *reverse* pada program yang lain. Jika dipaksakan juga untuk menggunakan fungsi *reverse*, maka cara yang ditempuh adalah menuliskan kembali fungsi *reverse* pada program yang baru. Hal ini akan menjadi masalah jika fungsi yang ditulis kembali berjumlah banyak, dan ini akan meningkatkan resiko kesalahan.

#### 2.2.1 Reusable Function

Cara yang baik untuk menyelesaikan problem sebelumnya adalah dengan membuat fungsi *reverse* terpisah dari program utama, kemudian mengkompilasinya menjadi modul terpisah, yang siap digabungkan dengan program apapun yang membutuhkan modul tersebut. Konsep seperti ini yang dikenal dengan istilah *reusable function*.

#### 2.2.2 Membuat Reusable Function

Untuk membuat *reusable function*, langkah pertama yang harus dibuat adalah menyiapkan *header file* (\*.h) yang mengandung *prototype* dari fungsi, dan *source file* (\*.c) yang mengandung implementasi dari fungsi. Kemudian kedua *file* ini dikompilasi menjadi satu *object module* (\*.o) dengan menggunakan option "-c".

```
//mReverse.h
void reverse (char *before, char *after);

//mReverse.c
#include <stdio.h>
#include "mReverse.h"

void reverse(char *before, char *after) {
    int i;
    int j;
    int len;

    len=strlen(before);

    for(j=len-1,i=0;j>=0;j--,i++)
        after[i]=before[j];

    after[len] = NULL
}

//mainMReverse.c

#include <stdio.h>
#include "mReverse.h"

int main() {
    char str[100];

    reverse("cat",str);
    printf("reverse (\"cat\") = %s\n",str);
}
```

## 2.2.3 Kompilasi Program Dengan Modul Banyak

### 2.2.3.1 Membuat *Object Module*

```
$gcc -c mReverse.c      .. akan tercipta mReverse.o
$gcc -c mainMReverse.c .. akan tercipta mainMReverse.o
```

### 2.2.3.2 Melakukan Link antar *Object Module*

```
$gcc mReverse.o mainMReverse.o -o mainReverse.exe
```

Keluaran dari proses *link* antara object *module* mReverse.o dan mainMReverse.o adalah file "mainReverse.exe". Nama dari *output file* bisa diubah-ubah sesuai dengan kebutuhan, dan tidak harus diberikan tambahan '.exe'.

## 3. Pembuatan Pustaka Baru Turbo/Borland C untuk GCC

Pustaka baru Turbo/Borland C dibuat menggunakan konsep pemrograman modul banyak. Hal ini memberikan kemudahan untuk pengembangan berikutnya, di mana Pengembangan modul pustaka dapat dilakukan lebih leluasa tanpa mempengaruhi *source* program yang menggunakan modul tersebut.

Pustaka yang sudah dibuat sejauh ini baru merupakan pustaka sederhana yang masih membutuhkan pengembangan lebih lanjut. Ide pembuatan dari pustaka baru ini sangat sederhana, yaitu dengan membuat fungsi (seperti gotoxy, delay dll.) yang telah dikenal dalam Turbo/Borland C, menggunakan perintah-perintah yang terdapat dalam gcc.

Berikut ini adalah contoh pembuatan fungsi gotoxy yang telah dikenal dalam Turbo/Borland C. Perintah ini memiliki dua buah parameter yaitu x dan y, di mana x adalah kolom dan y adalah baris.

```
gotoxy(int x, int y)
```

Berbekal pengetahuan tersebut, kemudian dicoba untuk membuat fungsi dengan nama yang sama menggunakan perintah-perintah yang telah ada di gcc. Dalam gcc (pada pustaka curses.h) terdapat perintah move yang memiliki karakteristik yang hampir sama dengan perintah gotoxy, namun parameter yang dimiliki oleh kedua perintah ini, saling berkebalikan. Parameter pertama dari perintah move menunjukkan baris sedangkan pada perintah gotoxy menunjukkan kolom. Begitu juga dengan parameter kedua dari perintah move dan gotoxy juga saling berkebalikan. Sehingga fungsi baru yang dibuat adalah sebagai berikut :

```
void gotoxy (int x, int y)
{
    move(y,x);
}
```

## 4. Hasil Yang Telah Dicapai

### 4.1 Pustaka Yang Telah Dibuat

Terdapat dua buah pustaka dasar yang telah dibuat, yaitu conio dan dos. Dalam pustaka conio telah diimplementasikan fungsi gotoxy dan clrscr, sedangkan pada pustaka dos telah diimplementasikan fungsi delay.

```
//conio.h
#include <ncurses.h>
#define printf printf
#define scanf scanf
void gotoxy(int x, int y);
void clrscr(void);

//conio.c
#include "conio.h"
void gotoxy(int x, int y) { move(y,x); }
void clrscr(void) { clear(); }

//dos.h
void delay(int ms);

//dos.c
#include "dos.h"
void delay(int ms)
{
    refresh();
    delay_output(ms);
}
```

### 4.2 Kompilasi Pustaka

Kedua pustaka yang telah dibuat kemudian diubah menjadi *object module* dengan cara sebagai berikut :

```
$gcc -c conio.c
$gcc -c dos.c
```

Hasil dari proses di atas adalah dua buah *object module*, conio.o dan dos.o

### 4.3 Cara Memanfaatkan Pustaka

Berikut ini adalah contoh *source* yang menggunakan pustaka conio dan dos :

```
//mainConioDos.c
#include <stdio.h>
#include "conio.h"
#include "dos.h"
int main()
{
    //Hapus perintah berikut, jika dijalankan pada Turbo/Borland C
    initscr();
    int maju=1;
    int posX=2, posY=15;
    char nama[20];
    clrscr();
    printf("Tekan CTRL + C, untuk keluar Jika Menggunakan Gcc
!\\n");
    printf("Tekan CTRL + BREAK, untuk keluar Jika Menggunakan Turbo
C !\\n");
    printf("\\nNama anda :");
    scanf("%s",nama);
    while (1)
    {
        if (maju)
        {
            if(posX<78) posX++; else maju=0;
        } else
        {
            if(posX>1) posX--; else maju=1;
        }
        gotoxy(posX,posY);
        printf(" [[ Halo : %s ]] ", nama);
        delay(100);
    }
    //Hapus perintah berikut, jika dijalankan pada Turbo/Borland C
    endwin();

    return 0;
}
```

### 4.4 Penggabungan Main Program dan Modul Pustaka

Berikut ini adalah cara menggabungkan modul pustaka yang telah ada dengan main program :

- Langkah pertama : ubah main program menjadi *object module*  
\$gcc -c mainConioDos.c
- Langkah kedua : menggabungkan semua *object module* dan membuat *executable file*  
\$gcc conio.o dos.o mainConioDos.o -o mainConioDos.exe -lnurses

Penambahan option “-lnurses” disebabkan karena terdapat perintah di dalam pustaka ncurses yang digunakan. Untuk mempermudah proses kompilasi dan *linking*, bisa dibuat sebuah *script*. Setelah proses penggabungan ini, *executable file* mainConioDos.exe akan tercipta. Berikut ini adalah cara menjalankannya (dengan asumsi mainConioDos.exe berada pada direktori aktif) :

\$/mainConioDos.exe

### 5. Saran Pengembangan

Pustaka yang telah dibuat masih merupakan embrio, yang butuh pengembangan lebih lanjut. Pustaka yang telah ada bisa dikembangkan ke arah yang lebih sempurna sehingga bisa mendukung semua perintah mode teks yang terdapat di Turbo/Borland C, seperti textcolor, textbackground dll.

## 6. Kesimpulan

Pemrograman dengan menggunakan banyak modul, memberikan kemudahan dalam pengembangan *software*. Fungsi-fungsi yang sering digunakan dapat dibuat dalam bentuk modul terpisah sehingga jika dibutuhkan cukup melakukan proses *linking*.

Pustaka yang telah direalisasikan hanya merupakan suatu contoh pemanfaatan pemrogram modul banyak. Dengan metode yang sama, pustaka-pustaka lain dapat juga dikembangkan sesuai dengan kebutuhan, seperti pustaka yang berisi fungsi matematika.

## Daftar Pustaka

[1] <http://linux.about.com/cs/linux101/g/gcc.htm>

[2] Graham Glass, “*UNIX for Programmers and Users A Complete Guide*”, Prentice-Hall International, 1993

## BIOGRAFI PENULIS



**Victor Nommmeinya Papilaya**, Lahir di Ambon 20 Mei 1979. Menamatkan SMU di SMUK YSKI Semarang dan menyelesaikan program S1 di UKSW - Universitas Kristen Satya Wacana Salatiga Jateng program studi Teknik Elektro pada tahun 2004. Saat ini bekerja sebagai pengajar muda di Fakultas Teknik Elektro UKSW dan aktif sebagai pembina STC – *Simple Training Club* yang mengkhususkan pada pelatihan peningkatan *skill* di bidang pemrograman dan jaringan komputer berbasis *open source*.