

JOB SHEET 11

File Handling in C

*"Everybody should learn to program a computer,
because it teaches you how to think."*

TUJUAN PEMBELAJARAN

1. Mampu menjelaskan dan mengimplementasikan *Write & Read dalam File* dalam pemrograman menggunakan IDE.

POKOK MATERI

1. Membaca File dengan C
2. Menulis dalam File dengan C

URAIAN MATERI

A. Pengantar

Apa yang akan terjadi bila program ditutup? Yap! semua data yang tersimpan di variabel akan hilang. Memanfaatkan variabel saja tidak akan cukup untuk menyimpan data. Karena itu, kita membutuhkan media penyimpanan lain seperti File. Dengan demikian, kita akan tetap memiliki data, meskipun programnya sudah ditutup.

Filenya juga bisa kita kirim ke komputer lain dan dibuka dari sana.

Kita akan belajar cara membaca dan menulis data ke file dengan bahasa C.

Membaca File dengan C

Sebelum kita mulai, saya ingin kasih tahu:

"File di komputer itu ada dua jenis, yakni file teks dan binary."

File teks biasanya dibuat dengan teks editor, contohnya seperti: file txt, file csv, file html, dll. File teks mudah dibaca dan ditulis.

Sedangkan file binary adalah file yang tersimpan dalam bentuk biner (0 & 1). Contohnya seperti: File exe dan file bin. File binary sulit dibaca, namun dapat menyimpan data lebih banyak dan aman.

Berikut ini adalah fungsi untuk membuka atau membaca file di C:



```
fopen("filepath", "r");
```

Fungsi fopen() akan membuka file sesuai dengan mode yang kita berikan.

Mode r artinya read atau baca saja. Selain mode r ada juga mode yang lain.

Berikut ini daftar lengkap modenya:

Mode	Arti	Jika File Tidak ada
r	Buka untuk dibaca	return NULL.
rb	Buka untuk dibaca dalam binary.	return NULL.
w	Buka untuk ditulis	Buat baru atau tulis ulang jika ada
wb	Buka untuk ditulis dalam binary	Buat baru atau tulis ulang jika ada

Mode	Arti	Jika File Tidak ada
a	Buka untuk ditambahkan	Buat baru atau tulis ulang jika ada
ab	Buka untuk ditambahkan dalam binary	Buat baru atau tulis ulang jika ada
r+	Buka untuk dibaca dan ditulis	returns NULL.
rb+	Buka untuk dibaca dan ditulis dalam binary	returns NULL.
w+	Buka untuk ditulis dan dibaca	Buat baru atau tulis ulang jika ada
wb+	Buka untuk dibaca dan ditulis dalam binary	Buat baru atau tulis ulang jika ada
a+	Buka untuk dibaca dan ditambahkan isinya	Buat baru atau tulis ulang jika ada
ab+	Buka untuk dibaca dan ditambahkan isinya dalam binary	Buat baru atau tulis ulang jika ada

Fungsi `fopen()` akan menghasilkan sebuah pointer yang menunjuk ke alamat memori dari file yang akan dibuka, karena itulah kita membutuhkan pointer untuk mengaksesnya.

```
// membuat pointer
File *fptr;

// membuka file
fptr = fopen("namafile.txt", "r");
```

Buatlah program baru bernama `baca_file.c`, kemudian isi dengan kode berikut:

```
#include <stdio.h>

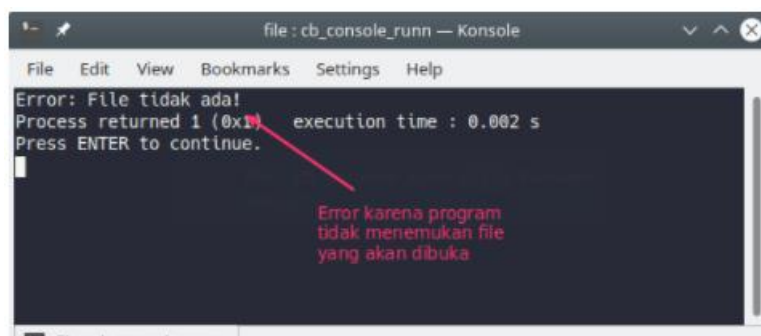
void main()
{
    char buff[255];
    FILE *fptr;

    // membuka file
    if ((fptr = fopen("puisi.txt", "r")) == NULL){
        printf("Error: File tidak ada!");
        // Tutup program karena file gak ada.
        exit(1);
    }

    // baca isi file dengan gets lalu simpan ke buff
    fgets(buff, 255, fptr);
    // tampilkan isi file
    printf("%s", buff);

    // tutup file
    fclose(fptr);
}
```

Setelah itu, coba jalankan programnya.



Hasilnya akan error, karena program tidak menemukan file yang akan dibuka.

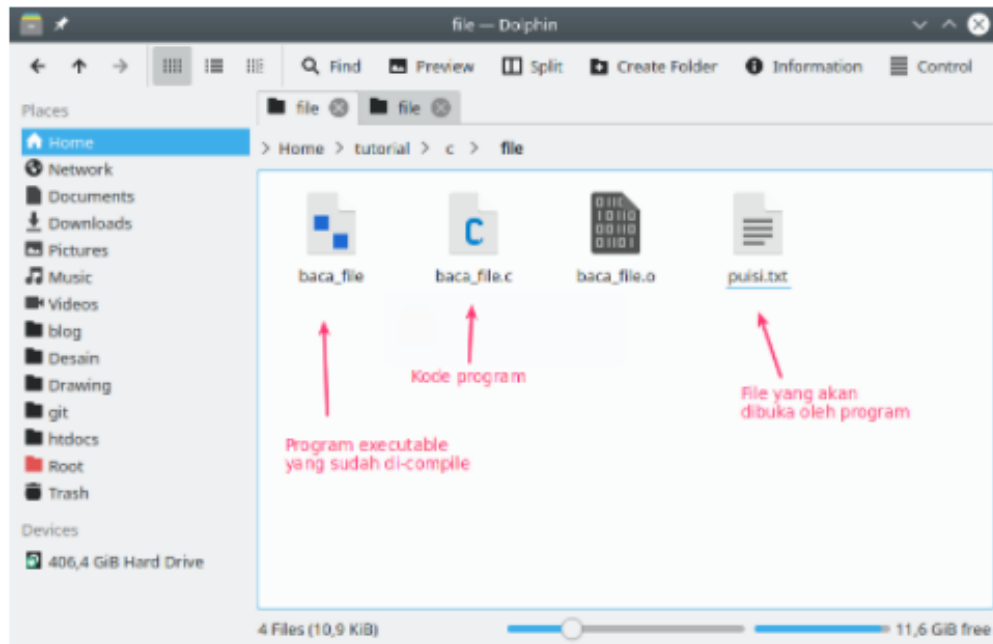
Sekarang coba buat file puisi.txt dengan isi sebagai berikut:

```
Ini adalah puisi untuk kamu
tapi isinya gak jelas

Ah yang penting file ini
ada isinya.

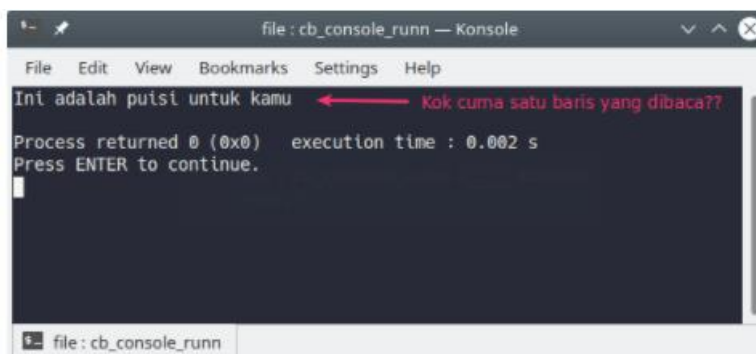
Cuma buat ngetes aja sih.
```

Simpan file ini satu direktori dengan programnya.



Setelah itu, coba jalankan lagi programnya.

Maka hasilnya:



Isi file akan dibaca satu baris karena kita menggunakan fungsi `fgets()`.

```
fgets(buff, 255, fptr);
```

Fungsi `fgets()` akan membaca isi file yang ditunjuk oleh pointer `fptr`, kemudian hasilnya akan disimpan ke dalam variabel `buff`.

Fungsi `fgets()` hanya akan membaca satu baris saja. Angka 255 adalah panjang baris (karakter) yang akan dibaca.

Jika kita ubah menjadi seperti ini:

```
fgets(buff, 20, fptr);
```

Maka hasil outputnya:

```
Ini adalah puisi un
```

Panjang karakter ini, bisa juga kita ambil dengan fungsi `sizeof()` agar mengikuti panjang karakter dari variabelnya

Contoh:

```
sizeof(buff);
```

Lalu bagaimana cara membaca semua baris teks yang ada di dalam file?

Gampang!

Kita tinggal panggil fungsi `fgets()` berkali-kali.

Sekarang coba ubah kode programnya menjadi seperti ini:

```
#include <stdio.h>

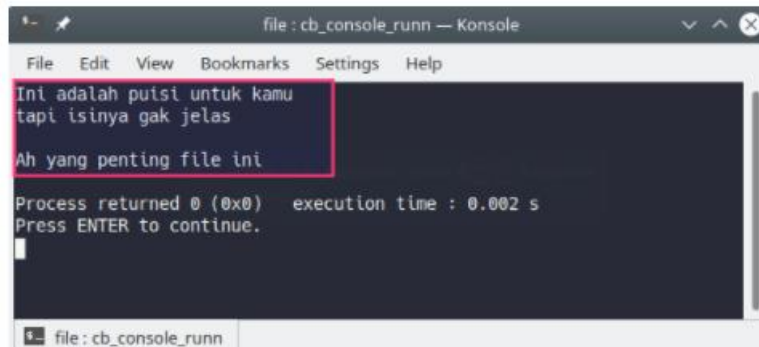
void main()
{
    char buff[255];
    FILE *fptr;

    // membuka file
    if ((fptr = fopen("puisi.txt", "r")) == NULL){
        printf("Error: File tidak ada!");
        // Tutup program karena file gak ada.
        exit(1);
    }

    // baca isi file dengan gets lalu simpan ke buff
    fgets(buff, sizeof(buff), fptr);
    printf("%s", buff);
    fgets(buff, sizeof(buff), fptr);
    printf("%s", buff);
    fgets(buff, sizeof(buff), fptr);
    printf("%s", buff);
    fgets(buff, sizeof(buff), fptr);
    printf("%s", buff);

    // tutup file
    fclose(fptr);
}
```

Hasilnya:



Program akan membaca empat baris dari isi file, ini karena kita memanggil fungsi `fgets()` sebanyak empat kali.

Tapi masalahnya nanti:

“Gimana kalau isi filenya ada banyak, misal 1000 baris?”

“Kita gak mungkin kan harus mengetik perintah itu berulang-ulang”

Benar sekali

Ini bisa kita atasi dengan perulangan.

Contohnya seperti ini:

```
#include <stdio.h>

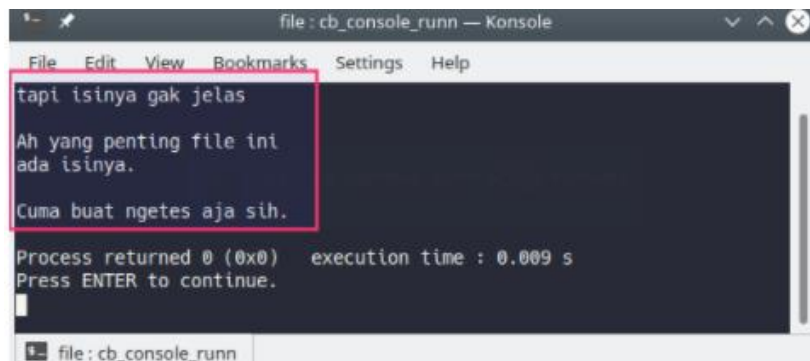
void main()
{
    char buff[255];
    FILE *fptr;

    // membuka file
    if ((fptr = fopen("puisi.txt", "r")) == NULL){
        printf("Error: File tidak ada!");
        // Tutup program karena file gak ada.
        exit(1);
    }

    // baca isi file dengan gets lalu simpan ke buff
    while(fgets(buff, sizeof(buff), fptr)){
        printf("%s", buff);
    }

    // tutup file
    fclose(fptr);
}
```

Maka hasilnya:



```
file: cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
tapi isinya gak jelas
Ah yang penting file ini
ada isinya.
Cuma buat ngetes aja sih.
Process returned 0 (0x0) execution time : 0.009 s
Press ENTER to continue.
```

Semua isi file akan dibaca. Ini karena kita memanggil fungsi `fgets()` di dalam perulangan `while`. Perulangan `while` akan berhenti saat `fgets()` menghasilkan `null` atau sudah tidak ada lagi baris yang dibaca.

Menulis dalam File dengan C

Jika fungsi `fgets()` digunakan untuk membaca file, nah untuk menulisnya kita butuh fungsi `fputs()`.

Kumpulan karakter
atau teks yang akan ditulis

pointer file

```
fputs('teks', fptr);
```

Fungsi `fputs()` akan menulis teks ke dalam file yang sedang dibuka.

Buatlah program baru dengan nama `tulis_file.c`, kemudian isi dengan kode berikut:


```
#include <stdio.h>

void main()
{
    char buff[255];
    char text[255];
    FILE *fptr;

    // membuka file
    fptr = fopen("puisi.txt","w");

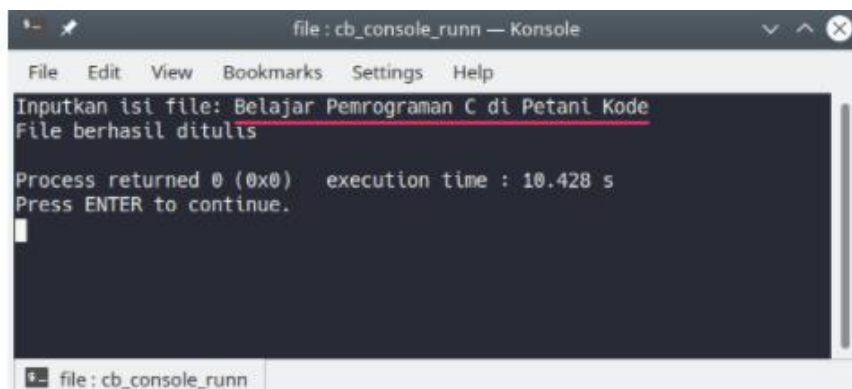
    // mengambil input dari user
    printf("Inputkan isi file: ");
    fgets(text, sizeof(text), stdin);

    // menulis ke text ke file
    fputs(text, fptr);

    printf("File berhasil ditulis\n");

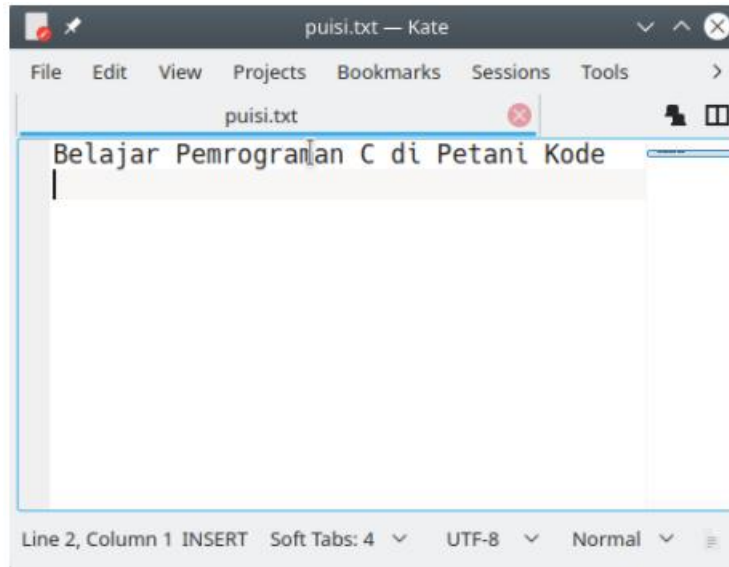
    // tutup file
    fclose(fptr);
}
```

Hasilnya:



```
file : cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
Inputkan isi file: Belajar Pemrograman C di Petani Kode
File berhasil ditulis
Process returned 0 (0x0)   execution time : 10.428 s
Press ENTER to continue.
file : cb_console_runn
```

Maka sekarang isi file puisi.txt akan seperti ini:

**Perhatikan!**

Jika kamu membuka file untuk ditulis, maka mode yang digunakan adalah w atau w+, a. Silahkan cek kembali tabel mode di atas. Kalau salah pakai mode gimana?

Ya filenya tidak akan bisa ditulis.

Lalu, gimana cara menulis dan sekaligus menampilkan isi file?

Caranya, kita harus membuka ulang filenya dengan mode yang berbeda-beda.

Contoh:

```
#include <stdio.h>

void main()
{
    char buff[255];
    char text[255];
    FILE *fptr;

    // membuka file untuk ditulis
    fptr = fopen("puisi.txt", "w");

    // mengambil input dari user
    printf("Inputkan isi file: ");
    fgets(text, sizeof(text), stdin);

    // menulis ke text ke file
    fputs(text, fptr);

    printf("File berhasil ditulis\n");
}
```

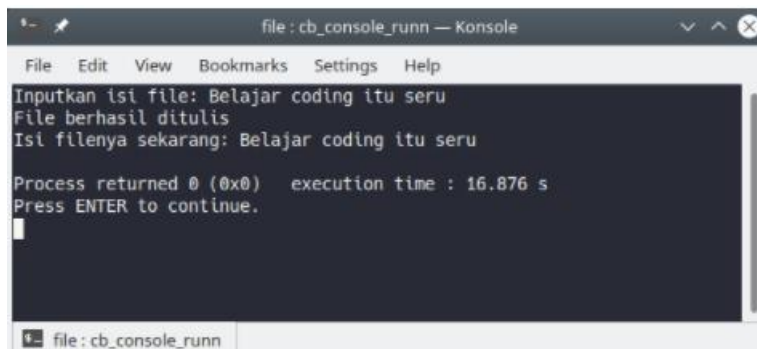
```
// tutup file setelah ditulis
fclose(fptr);

// buka kembali file untuk dibaca
fptr = fopen("puisi.txt","r");

// baca isi file dengan gets lalu simpan ke buff
while(fgets(buff, sizeof(buff), fptr)){
    printf("Isi filenya sekarang: %s", buff);
}

// tutup file
fclose(fptr);
}
```

Hasilnya:



```
file: cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
Inputkan isi file: Belajar coding itu seru
File berhasil ditulis
Isi filenya sekarang: Belajar coding itu seru
Process returned 0 (0x0) execution time : 16.876 s
Press ENTER to continue.
```

Kita hanya baru menulis dalam satu baris saja. Lalu gimana kalau mau menulis banyak baris? Gampang, kita hanya tinggal memanggil fungsi `fputs()` berkali-kali.

Contoh:

```
#include <stdio.h>

void main()
{
    char buff[255];
    char text[255];
    FILE *fptr;

    // membuka file untuk ditulis
    fptr = fopen("puisi.txt","w");

    for(int i = 0; i < 5; i++){

        // mengambil input dari user
        printf("Isi baris ke-%d: ", i);
        fgets(text, sizeof(text), stdin);

        // menulis ke text ke file
        fputs(text, fptr);
    }

    printf("File berhasil ditulis\n");

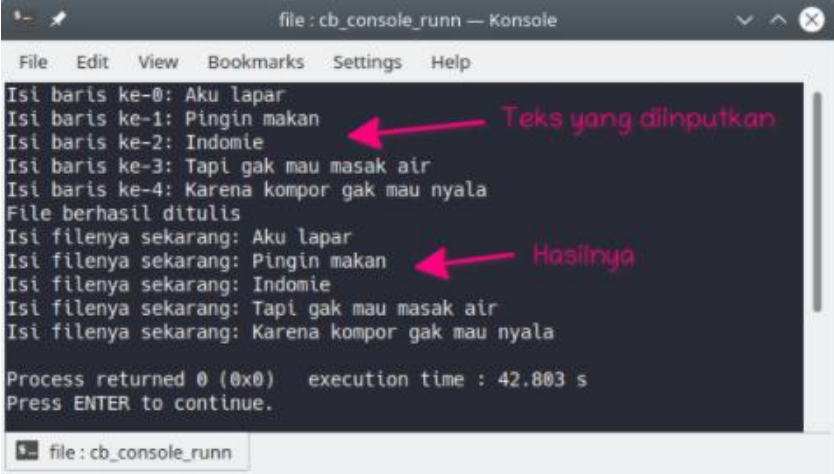
    // tutup file setelah ditulis
    fclose(fptr);

    // buka kembali file untuk dibaca
    fptr = fopen("puisi.txt","r");

    // baca isi file dengan gets lalu simpan ke buff
    while(fgets(buff, sizeof(buff), fptr)){
        printf("Isi filenya sekarang: %s", buff);
    }

    // tutup file
    fclose(fptr);
}
```

Maka hasilnya:



The screenshot shows a console window titled "file : cb_console_runn — Konsole". The output text is as follows:

```
Isi baris ke-0: Aku lapar
Isi baris ke-1: Pingin makan
Isi baris ke-2: Indomie
Isi baris ke-3: Tapi gak mau masak air
Isi baris ke-4: Karena kompor gak mau nyala
File berhasil ditulis
Isi filenya sekarang: Aku lapar
Isi filenya sekarang: Pingin makan
Isi filenya sekarang: Indomie
Isi filenya sekarang: Tapi gak mau masak air
Isi filenya sekarang: Karena kompor gak mau nyala

Process returned 0 (0x0)   execution time : 42.803 s
Press ENTER to continue.
```

Two handwritten pink annotations are present:

- A pink arrow points from the text "Teks yang diinputkan" to the line "Isi baris ke-1: Pingin makan".
- A pink arrow points from the text "Hasilnya" to the line "Isi filenya sekarang: Pingin makan".

The status bar at the bottom of the window shows "file : cb_console_runn".

LATIHAN

1. Buatlah program sederhana untuk menginputkan beberapa biodata mahasiswa.

(Gunakan perulangan)

Masukan data mahasiswa ke-1

Nama :

NIM :

Jurusan :

Program Studi :

Masukan data mahasiswa ke-2

Nama :

NIM :

Jurusan :

Program Studi :

...

Kemudian setiap hasil inputan tersebut akan disimpan ke dalam sebuah file dengan nama file: **datamahasiswa.txt**

2. Bukalah file **datamahasiswa.txt** yang sudah berisikan inputan dari latihan 1 menggunakan program C, sertakan screenshot keluaran!

REFERENCES

1. The C Programming Language. 2nd Edition
2. [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))
3. https://en.wikipedia.org/wiki/Imperative_programming
4. <https://www.petanikode.com/tutorial/c/>
5. <https://www.learn-c.org/>
6. <https://www.tutorialspoint.com/cprogramming/index.htm>
7. <https://www.programiz.com/>
8. <https://www.dicoding.com/>
9. <https://data-flair.training/blogs/c-tutorials-home/>