

PEMROSESAN FILE

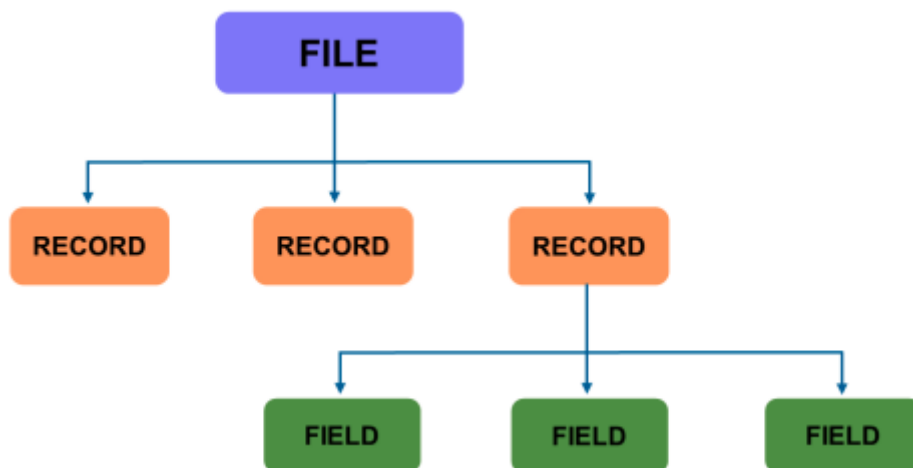
OBJEKTIF :

1. Mahasiswa mampu memahami tentang tahapan operasi file, operasi penyimpanan serta pembacaan data per karakter.
2. Mahasiswa mampu memahami tentang menyimpan dan membaca data string pada file dan data yang diformat.
3. Mahasiswa mampu memahami tentang menghapus dan mengganti nama file.

7.1 FILE

7.1.1 STRUKTUR FILE

Kebanyakan program melibatkan media disk sebagai tempat untuk membaca atau merekam data. Data sendiri disimpan dalam disk dalam bentuk suatu unit yang dinamakan file. Suatu file merupakan organisasi sejumlah `record`. Setiap `record` terdiri atas satu atau beberapa `field`, dan setiap `field` tersusun atas satu atau sejumlah `byte`. Adapun satu `byte` mengandung 8 `bit`. Berikut adalah struktur atau susunan dari file:



Dimana sebuah file terdiri dari beberapa `record` serta sebuah `record` terdiri dari beberapa `field`. Berikut adalah contoh dari `record` dan `field`:

Nama	Alamat	Kota
Andi	Jl. Apel	Jakarta
Kayla	Jl. Nangka	Depok
Santi	Jl. Jeruk	Bogor

→ Record

Field

Dimana masing - masing baris merupakan sebuah `record`. Pada tabel di atas terdapat 3 buah baris yang berarti tabel tersebut memiliki 3 buah `record`. Sedangkan nilai dari masing - masing kolom merupakan sebuah `field`.

7.1.2 TAHAPAN OPERASI FILE

Pada dasarnya, operasi pada file terdiri dari tiga tahapan berikut:

- Membuka/mengaktifkan file
- Melaksanakan proses terhadap file
- Menutup file

Sebelum file dapat diakses (dibaca atau ditulis), file perlu dibuka terlebih dahulu. Untuk membuka suatu file, digunakan fungsi `fopen()`. Prototipe fungsi `fopen()` terdapat pada library `stdio.h`. Berikut adalah contoh deklarasi dari fungsi `fopen()`:

```
FILE *fopen(char *namafile, char *mode);
```

nama file yang akan dibuka jenis operasi yang akan dilakukan terhadap file

Jenis operasi file dapat berupa salah satu diantara mode - mode berikut ini:

Mode	Keterangan
r	Menyatakan file hanya akan dibaca. Dalam hal ini, file yang akan diproses harus terdapat di disk.
w	Menyatakan bahwa file baru akan diciptakan. Selanjutnya, operasi yang akan dilakukan terhadap file adalah operasi perekaman data. Seandainya file tersebut sudah ada di disk, isi semula akan terhapus.
a	Untuk membuka file yang sudah ada dan operasi yang akan dilakukan adalah operasi penambahan data.
r+	Untuk membuka file yang sudah ada dan operasi yang akan dilakukan berupa pembacaan serta penulisan data dari/ke file.
w+	Untuk membuka file dengan tujuan untuk pembacaan atau penulisan. Jika file sudah ada, isinya akan dihapus.
a+	Untuk membuka file, dengan operasi yang dapat dilakukan berupa perekaman maupun pembacaan. Jika file sudah ada, isinya tidak akan dihapus.

Nilai balik fungsi `fopen()` berupa pointer yang menunjuk ke tipe `FILE`, yaitu alamat memori dari file yang akan dibuka. Jika nilai balik berupa `NULL`, berarti operasi pembukaan file tidak berhasil dilakukan. Berikut adalah contoh pemakaian fungsi `fopen()`:

```
pf = fopen("coba.txt", "w");
```

Dengan variabel `pf` dideklarasikan sebagai berikut:

```
FILE *pf;
```

Pointer ke FILE yang berisi nilai balik `fopen()`

Maksud statement `pf = fopen("coba.txt", "w");` adalah sebagai berikut:

- Menciptakan dan membuka file bernama `coba.txt`
- Mode yang digunakan adalah `"w"` (mode penulisan ke file)
- Pointer ke FILE akan ditempatkan ke variabel Pointer `pf`

Dengan adanya instruksi di atas, seandainya file `coba.txt` sudah ada di disk, isi file tersebut akan hilang (data lama akan terhapus).

Apabila suatu file tidak diproses lagi, maka file harus ditutup. Terutama jika melakukan pemrosesan file yang jumlahnya lebih dari satu. Alasannya karena adanya keterbatasan jumlah file yang dibuka secara serentak atau bersamaan. Untuk menutup file, fungsi yang digunakan adalah `fclose()`. Bentuk deklarasinya adalah sebagai berikut:



```
int fclose(FILE *pf)
```

Saat pemanggilan fungsi `fclose()`, `pf` haruslah berupa variabel pointer bertipe `FILE` yang digunakan dalam pengaktifan file. Fungsi `fclose()` menghasilkan nilai balik berupa nol jika operasi penutupan file berhasil dilakukan.

Selain `fclose()`, terdapat pula fungsi bernama `fcloseall()` yang digunakan untuk menutup semua file yang sedang terbuka. Fungsi `fcloseall()` menghasilkan nilai `EOF` (`EOF` didefinisikan pada library `stdio.h`, yaitu bernilai `-1` jika terjadi kegagalan. Jika file berhasil ditutup, nilai balik fungsi ini berupa jumlah file yang ditutup. Bentuk deklarasinya adalah sebagai berikut:

```
int fcloseall(void);
```

Jika operasi `fputc()` berjalan dengan sempurna, nilai balik fungsi sama dengan nilai `kar`. Bila tidak berhasil melaksanakan penyimpanan, nilai balik fungsi berupa `EOF` (atau `-1`).

Berikut adalah contoh program dengan menggunakan fungsi `fopen()` dan `fclose()`:

```
#include <stdio.h>

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

    // membuka file
    if ((fptr = fopen("nama mahasiswa.txt", "r")) == NULL){
        printf("Error: File tidak ditemukan!");
        // 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);
}
```

Output program:

```
Error: File tidak ditemukan!  
Process returned 1 (0x1)   execution time : 0.064 s  
Press any key to continue.
```

jika file "nama mahasiswa.txt" belum dibuat maka akan menampilkan output seperti di atas, namun jika file tersebut sudah dibuat maka akan menampilkan output seperti berikut ini:

```
Nama Mahasiswa yang bersangkutan adalah sebagai berikut:  
  
Process returned 0 (0x0)   execution time : 0.199 s  
Press any key to continue.
```

Berikut adalah contoh program dengan menggunakan fungsi `fopen()` dengan file yang sudah dibuat dan menampilkan seluruh isi file:

```
#include <stdio.h>  
  
void main()  
{  
    char buff[255];  
    FILE *fptr;  
  
    // membuka file  
    if ((fptr = fopen("nama mahasiswa.txt", "r")) == NULL){  
        printf("Error: File tidak ditemukan!");  
        // Tutup program karena file gak ada.  
        exit(1);  
    }  
  
    // baca isi file dengan gets lalu simpan ke buff  
    fgets(buff, sizeof(buff), fptr);  
    // tampilkan isi file  
    printf("%s", buff);  
  
    // baca isi file dengan gets lalu simpan ke buff  
    fgets(buff, sizeof(buff), fptr);  
    // tampilkan isi file  
    printf("%s", buff);  
  
    // baca isi file dengan gets lalu simpan ke buff  
    fgets(buff, sizeof(buff), fptr);  
    // tampilkan isi file  
    printf("%s\n", buff);  
  
    // tutup file  
    fclose(fptr);  
}
```

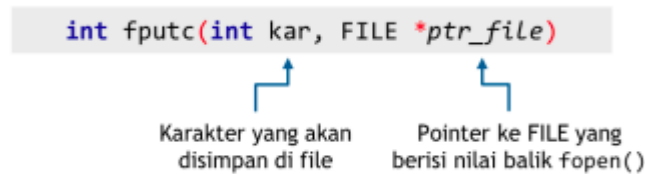
Output program:

```
Nama Mahasiswa yang bersangkutan adalah sebagai berikut:  
Raden Ayu Putri Ningsih  
Process returned 0 (0x0)   execution time : 0.899 s  
Press any key to continue.
```

7.1.3 OPERASI PENYIMPANAN DAN PEMBACAAN DATA PER KARAKTER

7.1.3.1 Fungsi `fputc()`

Sebuah karakter dapat disimpan di file dengan menggunakan fungsi `fputc()`. Bentuk deklarasi fungsinya adalah sebagai berikut:



Jika operasi `fputc()` berjalan dengan sempurna, nilai balik fungsi sama dengan nilai `kar`. Bila tidak berhasil melaksanakan penyimpanan, nilai balik fungsi berupa `EOF` (atau `-1`).

Berikut adalah contoh program dengan menggunakan fungsi `fputc()`:

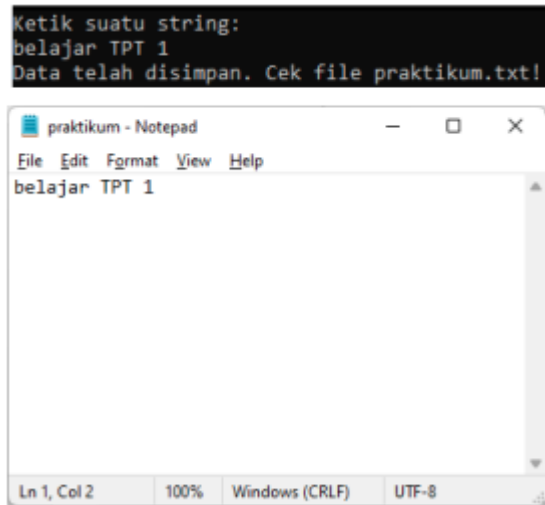
```
#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
    FILE *pf;  
    char kar;  
    char penampung[256];  
    int indeks;  
  
    pf = fopen("praktikum.txt", "w");  
    if (pf == NULL)  
    {  
        puts("File tidak dapat dibuat!");  
        exit(1);  
    }  
  
    printf("ketik suatu string: \n");  
    gets(penampung);  
    indeks = 0;  
    while (penampung[indeks] != 0)  
    {  
        kar = penampung[indeks];  
        fputc(kar,pf);  
        indeks++;  
    }  
  
    fclose(pf);  
}
```

```
printf("Data telah disimpan. Cek file praktikum.txt!\n");

return 0;

}
```

Output program:



7.1.3.2 Fungsi `getc()`

Fungsi `getc()` digunakan untuk membaca satu karakter yang ada di file. Prototipe fungsi `getc()` terdapat pada library `stdio.h`. Nilai balik fungsi `getc()` adalah nilai bertipe `int` yang menyatakan nilai karakter yang dibaca. Jika ditemukan atau terjadi kegagalan dalam membaca suatu file, nilai balik fungsi berupa `EOF`. Bentuk deklarasi fungsi `getc()` adalah sebagai berikut:

```
int getc(FILE *ptr_file)
```

↑
Pointer ke FILE yang berisi
nilai balik `fopen()`

Jika operasi `fputc()` berjalan dengan sempurna, nilai balik fungsi sama dengan nilai `kar`. Bila tidak berhasil melaksanakan penyimpanan, nilai balik fungsi berupa `EOF` (atau `-1`).

Berikut adalah contoh program dengan menggunakan fungsi `getc()`:

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

int main()
{
    FILE *pf;

    pf = fopen("praktikum.txt", "r");
    if (pf == NULL)
    {
        puts("File tidak dapat dibuka!");
        exit(1);
    }
    char kar = getc(pf);
```

```

while (kar != EOF)
{
    putchar(kar);
}

fclose(pf);

return 0;
}

```

Output program:

```
belajar TPT 1
```

7.2 MENYIMPAN DAN MEMBACA DATA

7.2.1 MENYIMPAN DAN MEMBACA DATA STRING PADA FILE

Dua fungsi yang digunakan untuk membaca dan menyimpan data string pada file yaitu `fgets()` dan `fputs()`.

7.2.1.1 Fungsi `fgets()`

Fungsi `fgets()` digunakan Untuk membaca string dari file sampai ditemukannya karakter baris baru (newline) atau setelah n-1 karakter (n adalah panjang maksimal string yang dibaca dalam sekali baca). Bentuk deklarasi fungsi `fgetc()` adalah sebagai berikut:

```
char *fgets(char *str, int n, FILE *ptr_file);
```



Nilai balik fungsi `fgets()` dapat berupa:

- Jika pembacaan data dari file berhasil, hasilnya berupa `pointer` yang menunjuk `string` yang ditunjuk oleh `str`
- Jika pembacaan data gagal, hasilnya berupa `NULL`

Berikut adalah contoh program dengan menggunakan fungsi `fgetc()`:

```

#include <stdio.h>
#define PANJANG 256

int main()
{
    FILE *pf;
    char str[PANJANG];
    char namafile[65];

```

```

printf("Masukkan nama file: ");
gets(namafile);

pf = fopen(namafile, "r");
if (pf == NULL)
{
    puts("File tidak ditemukan!");
    exit(1);
}

while (fgets(str, PANJANG, pf) != NULL)
    printf(str);

fclose(pf);

return 0;
}

```

Output program:

```

Masukkan nama file: baca_file.txt
Membaca isi file menggunakan fungsi fgets()
Pada Bahasa Pemrograman C

```


7.2.1.2 Fungsi `fputs()`

Fungsi `fputs()` digunakan Untuk menyimpan `string` `str` ke dalam file. Bentuk deklarasi fungsi `fputs()` adalah sebagai berikut:

```

int fputs(char *str, FILE *ptr_file);

```



String/karakter yang akan disimpan ke dalam file

Pointer ke FILE yang berisi nilai balik fopen()

Nilai balik fungsi `fputs()` dapat berupa:

- Jika penyimpanan ke file berhasil, hasilnya berupa karakter yang terakhir ditulis ke file
- Jika pembacaan data gagal, hasilnya berupa `EOF`

Berikut adalah contoh program `salinfile.c`:

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define PANJANG 256

int main()
{
    FILE *pf_inp, *pf_out;
    char str[PANJANG];
    char namafile_inp[65], namafile_out[65];

```



```

printf("Program untuk menyalin file teks\n");
printf("Dalam hal ini, setiap huruf kecil\n ");
printf("diganti dengan huruf kapital\n");

printf("Nama file masukan: ");
gets(namafile_inp);

printf("Nama file keluaran: ");
gets(namafile_out);
pf_inp = fopen(namafile_inp, "r");
if (pf_inp == NULL)
{
    puts("File tidak ditemukan!");
    exit(1);
}

pf_out = fopen(namafile_out, "w");
if (pf_out == NULL)
{
    puts("File tidak ditemukan!");
    exit(1);
}

while (fgets(str, PANJANG, pf_inp) != NULL)
{
    strupr(str);
    fputs(str, pf_out);
}

fclose(pf_inp);
fclose(pf_out);

return 0;
}

```

Output program:

```
Program untuk menyalin file teks
Dalam hal ini, setiap huruf kecil
diganti dengan huruf kapital
Nama file masukan: salinfile.c
Nama file keluaran: salinfile.kap
```

```
C:\Windows\System32\cmd.exe
C:\lab\tp1\type salinfile.kap
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define PANJANG 256

int main()
{
    FILE *pf_inp, *pf_out;
    char str[panjang];
    char namafile_inp[65], namafile_out[65];

    printf("PROGRAM UNTUK MENYALIN FILE TEKS\n");
    printf("DALAM HAL INI, SETIAP HURUF KECIL\n");
    printf("DIGANTI DENGAN HURUF KAPITAL\n");

    printf("NAMA FILE MASUKAN: ");
    gets(namafile_inp);

    printf("NAMA FILE KELUARAN: ");
    gets(namafile_out);

    pf_inp = fopen(namafile_inp, "r");
    if (pf_inp == NULL)
    {
        puts("FILE TIDAK DAPAT DIBUKAI");
        exit(1);
    }

    pf_out = fopen(namafile_out, "w");
    if (pf_out == NULL)
    {
        puts("FILE TIDAK DAPAT DIBUKAI");
        exit(1);
    }

    while (fgets(str, panjang, pf_inp) != NULL)
    {
        strupr(str);
        fputs(str, pf_out);
    }

    fclose(pf_inp);
    fclose(pf_out);

    return 0;
}
```

7.2.2 MENYIMPAN DAN MEMBACA DATA YANG DIFORMAT

7.2.2.1 Fungsi `fprintf()`

Jika dikehendaki, data bilangan dapat disimpan ke file dalam keadaan diformat. Sebagai contoh, bilangan bulat bertipe `short int` disimpan dengan panjang lima digit dan dibuat rata kanan, berapapun nilainya Fungsi yang digunakan adalah `fprintf()`, dengan deklarasinya sebagai berikut:

```
fprintf(FILE *ptr_file, "string kontrol", daftar_argumen);
```



Berikut adalah contoh program dengan menggunakan fungsi `fprintf()`:

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

int main()
{
    FILE *pf;
    char jawaban;
```

```

struct
{
    int x;
    int y;
} koordinat;

pf = fopen("koordinat.dat", "w");
if (pf == NULL)
{
    puts("File tidak dapat dibuka!");
    exit(1);
}

do
{
    printf("Masukkan data koordinat (integer)\n");
    printf("Format: posisi_x posisi_y\n");
    printf("Contoh: 20 30 ENTER\n");

    scanf("%d %d", &koordinat.x, &koordinat.y);

    fflush(stdin);

    fprintf(pf, "%5d %5d\n", koordinat.x, koordinat.y);

    printf("Mau masukkan data lagi (Y/T)? ");
    jawaban = getchar();
} while (jawaban == 'Y' || jawaban == 'y');

fclose(pf);

return 0;
}

```

Output program:

```

Masukkan data koordinat (integer)
Format: posisi_x posisi_y
Contoh: 20 30 ENTER
8 34
Mau masukkan data lagi (Y/T)? Y
Masukkan data koordinat (integer)
Format: posisi_x posisi_y
Contoh: 20 30 ENTER
879 45
Mau masukkan data lagi (Y/T)? y
Masukkan data koordinat (integer)
Format: posisi_x posisi_y
Contoh: 20 30 ENTER
123 6798
Mau masukkan data lagi (Y/T)? y
Masukkan data koordinat (integer)
Format: posisi_x posisi_y
Contoh: 20 30 ENTER
23000 46575
Mau masukkan data lagi (Y/T)? t

```

7.2.2.2 Fungsi fscanf()

Untuk membaca kembali data yang ditulis melalui `fprintf()`, digunakan fungsi `fscanf()`. Deklarasi fungsi `fscanf()` adalah sebagai berikut:

```
fscanf(FILE *ptr_file, "string kontrol", daftar_argumen);
```



Berikut adalah contoh program dengan menggunakan fungsi `fscanf()`:

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

int main()
{
    FILE *pf;
    char kar;
    int kode;

    struct
    {
        int x;
        int y;
    } koordinat;

    pf = fopen("koordinat.dat", "r");
    if (pf == NULL)
    {
        puts("File tidak dapat dibuka!");
        exit(1);
    }

    while ((kode = fscanf(pf, "%5d%5d%c",
                        &koordinat.x,
                        &koordinat.y,
                        &kar)) != EOF)
    {
        if (kode != 3)
        {
            printf("Ada format yang salah!");
            exit(1);
        }

        printf("%5d %5d\n", koordinat.x, koordinat.y);
    }

    fclose(pf);

    return 0;
}
```

```
}
```

Output program:

```
8      34
879    45
123    6798
23000  46575
```

7.3 MENGHAPUS DAN MENGGANTI NAMA FILE

7.3.1 MENGHAPUS NAMA FILE

7.3.1.1 Fungsi `unlink()`

Untuk menghapus file, digunakan fungsi `unlink()`. Fungsi `unlink()` terdapat pada *library* `stdio.h`. Deklarasi fungsi `unlink()` adalah sebagai berikut :

```
int unlink(char *nama_file);
```

↑
Pointer yang menunjuk ke
nama file yang akan dihapus

Nilai balik fungsi `unlink()` dapat berupa:

- nol, jika operasi penghapusan file berhasil dilakukan
- -1, jika terjadi kegagalan

Berikut adalah contoh program untuk menghapus file dengan menggunakan fungsi `unlink()` :

```
#include <stdio.h>

int main()
{
    int kode;
    char namafile[65];

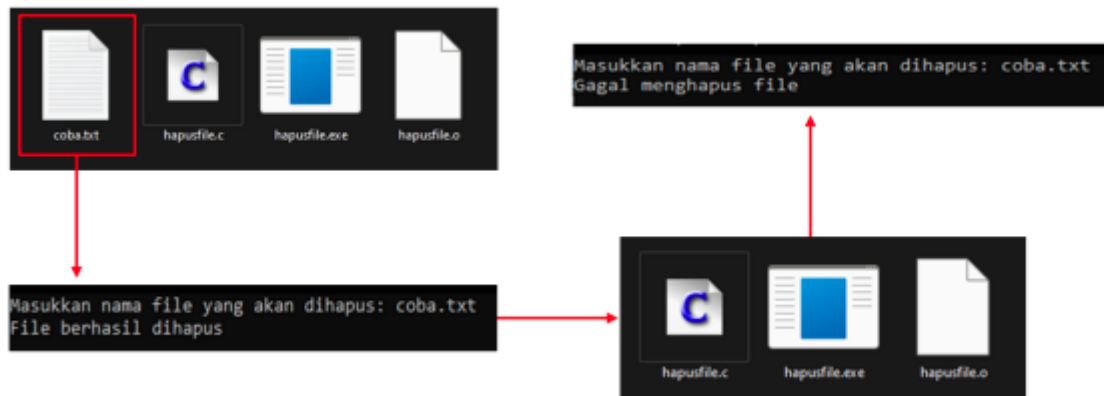
    printf("Masukkan nama file yang akan dihapus: ");
    gets(namafile);

    kode = unlink(namafile);

    if (kode == -1)
        printf("Gagal menghapus file\n");
    else
        printf("File berhasil dihapus\n");

    return 0;
}
```

Output program:



7.3.1.2 Fungsi `remove()`

Selain menggunakan fungsi `unlink()`, untuk menghapus file juga dapat menggunakan fungsi `remove()`. Deklarasi fungsi `remove()` mirip seperti fungsi `unlink()`, yaitu sebagai berikut:

```
int remove(char *nama_file);
```

↑
Pointer yang menunjuk ke
nama file yang akan dihapus

Nilai balik fungsi `remove()` dapat berupa:

- nol, jika operasi penghapusan file berhasil dilakukan
- -1, jika terjadi kegagalan

Berikut adalah contoh program untuk menghapus file dengan menggunakan fungsi `remove()`:

```
#include <stdio.h>

int main()
{
    if (remove("coba.txt") == 0)
        printf("Berhasil menghapus");
    else
        printf("Tidak dapat menghapus file tersebut");

    return 0;
}
```

Output program:

```
Berhasil Menghapus
Process returned 0 (0x0)   execution time : 0.862 s
Press any key to continue.
```

7.3.2 MENGGANTI NAMA FILE

Untuk mengganti nama file, digunakan fungsi `rename()`. Fungsi `rename()` terdapat pada library `stdio.h`. Deklarasi fungsi `rename()` adalah sebagai berikut:

```
int rename(char *nama_file_lama, char nama_file_baru);
```

↑
Pointer yang menunjuk ke nama file yang akan diganti nama nya

↑
Nama file baru

Nilai balik fungsi `rename()` dapat berupa:

- nol, jika operasi penggantian nama file berhasil dilakukan
- -1, jika terjadi kegagalan

Berikut adalah contoh program untuk mengganti nama file dengan menggunakan fungsi `rename()`:

```
#include <stdio.h>

int main()
{
    int kode;
    char namafile_lama[65], namafile_baru[65];

    printf("Masukkan nama file yang akan diganti: ");
    gets(namafile_lama);

    printf("Masukkan nama file pengganti: ");
    gets(namafile_baru);

    kode = rename(namafile_lama, namafile_baru);

    if (kode == -1)
        printf("Gagal mengganti nama file\n");
    else
        printf("Nama file berhasil diganti\n");

    return 0;
}
```

Output program:



REFERENSI

[1] Abdul Kadir. 2015. *From Zero to a Pro*. Yogyakarta. Andi