

# OPERASI FILE

**DASAR PEMROGRAMAN**

# TUJUAN

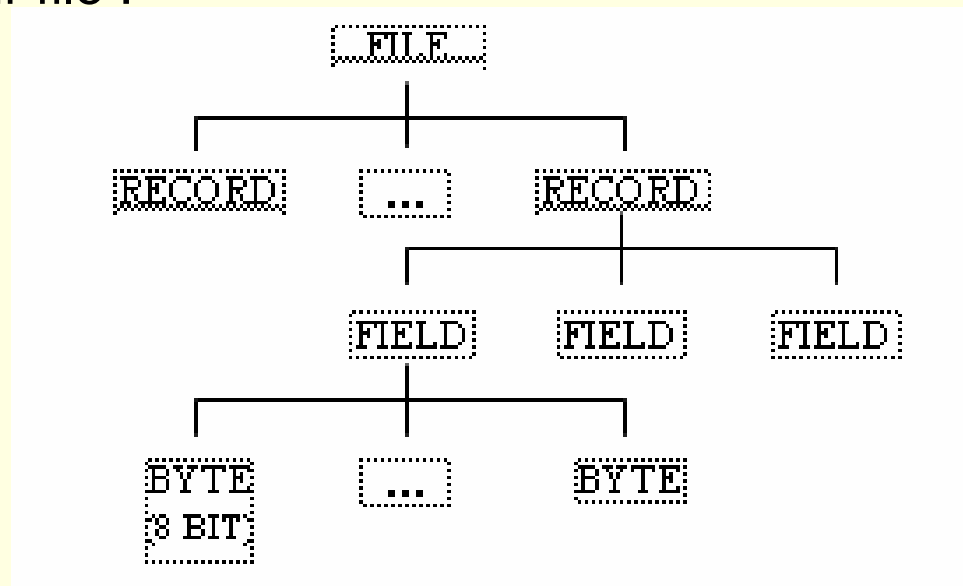
---

**Setelah menyelesaikan bab ini, mahasiswa diharapkan dapat:**

- **Menjelaskan tentang struktur file**
- **Menjelaskan tentang tahap-tahap operasi pada file**
- **Menjelaskan tentang fungsi untuk penyimpanan dan pembacaan file per-karakter**
- **Menjelaskan tentang file biner dan file teks**
- **Menjelaskan tentang operasi penyimpanan dan pembacaan file per-int**
- **Menjelaskan cara membaca dan menyimpan data string pada file**
- **Menjelaskan cara menghapus file**
- **Menjelaskan cara mengganti nama file**

# DASAR STRUKTUR FILE

- Penyimpanan suatu data dalam disk berupa suatu file.
- Gambar struktur file :

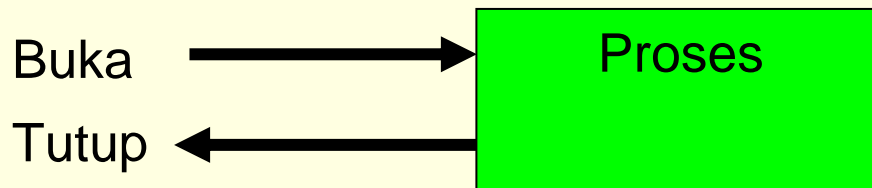


Catatan : *record* adalah nama lain dari struktur (*struct*).

# DASAR FILE – Cont. 1

---

- Tahapan Operasi File :
  1. Membuka/mengaktifkan file
  2. Melaksanakan proses file
  3. Menutup file



# MEMBUKA / AKTIFKAN FILE

---

- Bentuk deklarasi :

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

Keterangan :

- **namafile** berupa nama dari file yang akan diaktifkan
- **mode** berupa jenis operasi yang akan dilakukan terhadap file
- prototipe ada pada file **stdio.h**

# JENIS OPERASI FILE

---

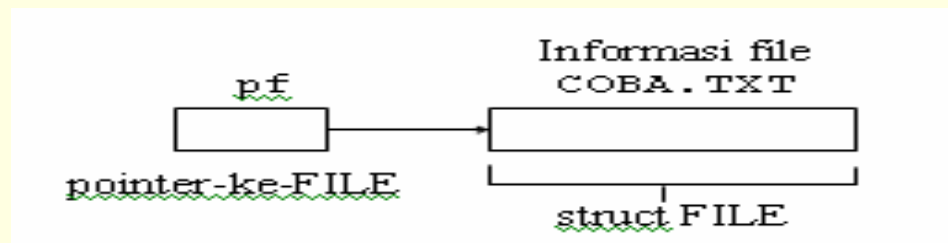
- **r** menyatakan file hanya akan dibaca, jika file belum ada maka tidak akan berhasil.
- **w** menyatakan bahwa file baru diciptakan. Jika file tersebut sudah ada dalam disk, isinya yang lama akan terhapus.
- **a** untuk membuka file yang sudah ada untuk ditambah dengan data, jika file belum ada akan dibuat yang baru.
- **r+** sama dengan “r” tetapi selain file dapat dibaca, file juga dapat ditulisi.
- **w+** sama dengan “w” tetapi selain file dapat ditulisi, file juga dapat dibaca.
- **a+** sama dengan “w” tetapi selain file dapat ditulisi, file juga dapat dibaca.

# JENIS OPERASI FILE – Cont. 1

- Berhasil tidaknya operasi pengaktifan file dapat dilihat pada keluaran fungsi *fopen()*.
- Jika keluaran fungsi berupa NULL (suatu makro yang didefinisikan pada file **stdio.h**), berarti operasi pengaktifan file gagal → misal membuka file dengan mode 'r' tapi filenya belum ada.
- Contoh :

```
FILE *pf; //deklarasi variabel pf  
pf = fopen("COBA.TXT", "w");
```

- ❑ menciptakan dan mengaktifkan file bernama "COBA.TXT"
- ❑ dengan mode yaitu "w" (mode penulisan ke file)
- ❑ dan menempatkan pointer-ke-FILE ke variabel pointer pf



# JENIS OPERASI FILE – Cont. 2

---

## ■ Contoh Bentuk pengaktifan file :

```
if (pf = fopen("COBA.TXT", "w") == NULL)
{
    printf("File tidak dapat diciptakan !\n");
    exit(1);           //keluar dari program
}
```

Keterangan :

- > pf akan diisi dengan keluaran dari fungsi *fopen()*.
- > Jika nilainya NULL, maka akan mencetak "File tidak dapat diciptakan", setelah itu program dihentikan.



# MENUTUP FILE

- Apabila file sudah tidak diproses lagi, maka file tersebut ditutup, karena adanya keterbatasan jumlah file yang dapat dibuka secara serentak.
- Perintah yang digunakan : `fclose()` ;
- Bentuk deklarasi :

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

- Bentuk deklarasi yang lain :

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

```
prototype yang digunakan : stdio.h
```

# OPERASI FILE

## Operasi Penyimpanan dan Pembacaan File Per Karakter

---

### A. OPERASI PENYIMPANAN FILE

- Penyimpanan karakter ke file menggunakan perintah : `fputc()`.
- Bentuk deklarasi :

```
int fputc(char kar, FILE *ptr_file);
```

- **ptr\_file** adalah pointer-ke-FILE yang berisi keluaran dari *fopen()*,
- kar berupa karakter yang akan disimpan dalam file.

# CONTOH PROGRAM TULIS

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

Bisa dilihat hasilnya dengan notepad

```
main()
{
```

```
    FILE *pf;           /* Pointer-ke-FILE */
    char kar;
```

```
    /* Ciptakan file */
```

```
    if ((pf = fopen("COBA.TXT","w")) == NULL)
    {
```

```
        printf("file tak dapat diciptakan!\r\n");
        exit(1);
    }
```

```
    //Masukkan karakter per karakter sampai ditekan ENTER
```

```
    while((kar=getchar()) != '\n')
        fputc(kar, pf);
```

```
    fclose(pf);         /* tutup file */
```

```
}
```

# OPERASI FILE – Cont. 1

---

## B. OPERASI PEMBACAAN FILE

- Pembacaan karakter dari suatu file memakai perintah : `fgetc()`.
- Bentuk deklarasi :

```
int fgetc(FILE *ptr_file);
```

Algoritma Proses Pembacaan File per karakter:

1. Buka file COBA.TXT dengan mode “r”

Jika tidak berhasil dibuka maka

- beri keterangan pada layar bahwa file tak ada
- selesai

2. Baca sebuah karakter dari file

Jika karakter sama dengan EOF (tanda akhir file) maka ke langkah 4

3. Tampilkan karakter ke layar dan kembali ke langkah 2

4. Tutup file

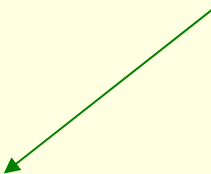
5. Selesai

# CONTOH PROGRAM BACA

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    FILE *pf;
    char kar;

    if((pf=fopen("COBA.TXT","r")) == NULL )           /* buka file */
    {
        printf("file tak dapat dibuka !\r\n");
        exit(1);
    }
    /*Baca karakter per karakter sampai ditemui EOF*/
    while((kar=fgetc(pf)) != EOF)
        putchar(kar);
    printf("\n");
    fclose(pf);                                         /* tutup file */
}
```

File ini sudah harus ada dulu di directory workspace C-nya



# CONTOH PROGRAM BACA TULIS

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

main()
{
    FILE *pf;                /* Pointer-ke-FILE */
    char kar;
    /* Ciptakan file */
    if ((pf = fopen("COBA.TXT", "r+")) == NULL)
    {
        printf("file tak dapat diciptakan!\r\n");
        exit(0);
    }
    while((kar=fgetc(pf)) != EOF) /* baca kar dari file */
        putchar(kar);

    while((kar=getchar()) != '\n') /* baca kar dr keyboard */
        fputc(kar, pf);

    fclose(pf);
}
```

# JENIS FILE

- **File Biner** : file yang pola penyimpanan di dalam disk berbentuk biner, yaitu seperti bentuk pada memori RAM (komputer).  
Dipakai untuk menyimpan data kompleks, mis : struct.
- **File Teks** : file yang pola penyimpanan datanya dalam bentuk karakter.  
Dipakai untuk menyimpan data seperti karakter atau string.

- **Penentuan mode teks dan mode biner :**

- t untuk mode teks

- b untuk mode biner

**Contoh :**

**"rt"** : mode file adalah teks dan file hendak dibaca

**"rt+"** : mode file adalah teks dan file bisa dibaca dan ditulisi.

Bisa juga ditulis : **"r+t"**

**"rb"** : mode file adalah biner dan file hendak dibaca.

# OPERASI BACA & TULIS

## FILE INTEGER

---

- Perintah yang digunakan : `_putw ()`, `_getw()`.
- Bentuk deklarasi :

```
int _putw(int nilai, FILE *ptr_file);  
int _getw(FILE *ptr_file);
```

Kegunaan :

`_getw()` untuk membaca sebuah data bertipe `int` dari file  
`_putw()` untuk menyimpan sebuah data bertipe `int` ke file.



# CONTOH PROGRAM TULIS

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
    FILE *pf;           /* ptr-ke-FILE */
    int nilai, data, i;
    char jawab;
    if((pf=fopen("BILANGAN.DAT", "wb")) == NULL )
    {
        printf("file gagal diciptakan!\n");
        exit(1);
    }
    printf ("Masukkan banyaknya data : ");
    scanf ("%d",&data);
    for (i=0;i<data;i++) {
        printf("\nBilangan yang disimpan: ");
        scanf("%d", &nilai);           /* baca nilai dr keyboard */
        _putw(nilai, pf);               /* baca bilangan ke file */
    }
    printf("\nOke. Data sudah disimpan dalam file.\n");
    fclose(pf);                        /* menutup file */
}
```

Masukkan banyaknya data : 3

Bilangan yang disimpan : 70

Bilangan yang disimpan : 80

Bilangan yang disimpan : 90

# CONTOH PROGRAM BACA

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    FILE *pf;           /* ptr ke file */
    int nilai, nomor = 0;
    /* Buka file biner untuk dibaca */
    if((pf=fopen("BILANGAN.DAT","rb")) == NULL)
    {
        printf("File gagal dibuka.\n");
        exit(1);
    }
    printf("Isi file BILANGAN.DAT : \n");
    while(1)             /* file berhasil dibuka */
    {
        nilai = _getw(pf); /* Baca sebuah int dr file */
        if (feof(pf) != 0) break; /* Jika akhir file, keluar loop */
        printf("%2d.  %d \n", ++nomor, nilai); /* Tampilkan ke layar */
    }

    fclose(pf);          /* Tutup file */
}
```


Isi file BILANGAN.DAT :

1. 70

2. 80

3. 90

feof : untuk  
mendeteksi  
akhir file



# OPERASI BACA & SIMPAN DATA STRING PADA FILE

- Perintah yang digunakan : *fgets()* dan *fputs()*.
- Bentuk deklarasi :

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

Kegunaan :

*fputs()* : menyimpan string *str* ke dalam file.

*fgets()* : membaca string dari file sampai ditemukannya karakter baris baru '\n' atau setelah (n-1) karakter, dengan n adalah panjang maksimal string yang dibaca per waktu-baca.

## Note :

- ❑ ***Saat simpan, fputs()* tidak menambahkan karakter baris-baru ('\n') dengan sendirinya, dan karakter null tidak ikut disimpan.**
- ❑ **Baik *fgets()* maupun *fputs()* digunakan untuk file teks.**

# CONTOH PROGRAM TULIS - 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main( )
{
    FILE *pf;           /* ptr-ke-FILE */
    int data, i;
    char nama[40];
    if((pf=fopen("latihan.txt", "w")) == NULL )
    {
        printf("file gagal diciptakan!\n");
        exit(1);
    }
    printf ("Masukkan banyaknya data : ");
    scanf ("%d",&data);
    for (i=1;i<=data;i++) {
        printf("\nNama ke %d : ",i); fflush(stdin);
        gets(nama);
        strcat(nama,"\n");
        fputs(nama, pf);
    }
    printf("\nOke. Data sudah disimpan dalam file.\n");
    fclose(pf);         /* menutup file */
}
```

# CONTOH PROGRAM BACA

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
    FILE *pf;           /* ptr-ke-FILE */
    int data, i;
    char nama[40];
    if((pf=fopen("latihan.txt", "r")) == NULL )
    {
        printf("file gagal dibuka!\n");
        exit(1);
    }
    /*Baca file per string sampai ditemui EOF*/
    while (fgets(nama,6,pf))
        printf ("%s\n",nama);

    fclose(pf);         /* menutup file */
}
```

Hanya  
mencetak 6  
karakter  
per baris

# MENGHAPUS FILE

---

- Bentuk deklarasi :

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

namafile : pointer yang menunjuk ke nama file yang akan dihapus

- Jika operasi hapus berhasil, akan menghasilkan output = 0.
- Prototype : stdio.h

# CONTOH PROGRAM HAPUS

---

```
#include <stdio.h>
#include <stdlib.h>
#define PJJ 65
main()
{
    int kode;
    char namafile[PJJ];

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

    kode = remove(namafile);
    if(kode == 0)
        printf("File sudah dihapus\n");
    else
        printf("Gagal dalam menghapus file\n");
}
```

# MENGGANTI NAMA FILE

---

- Bentuk deklarasi :

```
int rename(char *namafilelama, char *namafilebaru);
```

- ❑ Jika operasi penggantian berhasil, akan menghasilkan output = 0.
- ❑ Prototype : stdio.h



# CONTOH PROGRAM GANTI NAMA

```
#include <stdio.h>
#include <stdlib.h>
#define PJG 65
main()
{
    int kode;
    char namafilelama[PJG], namafilebaru[PJG];

    printf("Nama file yang akan diganti : ");
    gets(namafilelama);
    printf("Nama file yang baru      : ");
    gets(namafilebaru);

    kode = rename(namafilelama, namafilebaru);
    if(kode == 0)
        printf("Nama file sudah diganti\n");
    else
        printf("Gagal dalam mengganti nama\n");
}
```

# Latihan

---

- 1. Buatlah program untuk memasukkan sejumlah nama dan usia masing-masing ke dalam file bertipe teks.  
Lengkapi dengan program untuk membaca isi dari file yang sudah terbentuk**
- 2. Buat program untuk memasukkan 5 pasang bilangan bertipe integer ke dalam file bertipe biner. Selanjutnya lakukan operasi penjumlahan pada masing-masing pasangan, dan isikan hasil penjumlahan ke file yang sama.**