I File

Il file e` l'unita` logica di memorizzazione dei dati su memoria di massa.

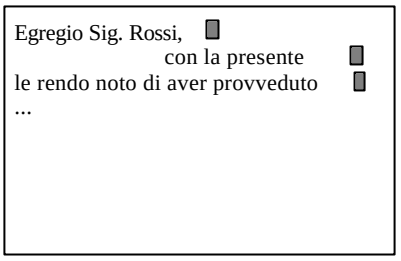
- Consente una ***memorizzazione persistente*** dei dati, ***non limitata*** dalle dimensioni della memoria centrale.
- Generalmente un file e` una sequenza di componenti omogenee (***record logici***).
- I file sono gestiti dal Sistema Operativo. Per ogni versione C esistono funzioni per il trattamento dei file (***Standard Library***) che tengono conto delle funzionalita` del S.O ospite.

In C i file vengono distinti in due categorie:

- ***file di testo***, trattati come sequenze di caratteri. organizzati in linee (ciascuna terminata da '\n')
- ***file binari***, visti come sequenze di bit

File di testo

Sono file di caratteri, organizzati in linee.
Ogni linea e` terminata da una marca di fine linea (***newline***, cara ttere '\n').

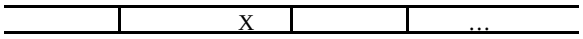


➳ Il ***record logico*** puo` essere il singolo carattere oppure la singola linea.

Gestione di file in C

I file hanno una struttura ***sequenziale:***

- i record logici sono organizzati in una sequenza
- per accedere ad un particolare record logico, e` necessario "scorrere" tutti quelli che lo precedono.



Per accedere ad un file da un programma C, e` necessario predisporre una variabile che lo rappresenti (**puntatore a file)**

Puntatore a file:

e` una variabile che viene utilizzata per riferire un file nelle operazioni di accesso (lettura e scrittura.). Implicitamente essa indica:

- il file
- l'elemento corrente all'interno della sequenza

Ad esempio:

```
FILE *fp;
```

➳ il tipo FILE e` un tipo non primitivo dichiarato nel file **stdio.h.**

Gestione di file in C

Apertura di un file:

Prima di accedere ad un file e` necessario ***aprirlo:*** l'operazione di apertura compie le azioni preliminari necessarie affinche` il file possa essere acceduto (in lettura o in scrittura). L'operazione di apertura inizializza il puntatore al file.

Accesso ad un file:

Una volta aperto il file, e` possibile leggere/scrivere il file, riferendolo mediante il puntatore a file.

Chiusura di un file:

Alla fine di una sessione di accesso (lettura o scrittura) ad un file e` necessario chiudere il file per memorizzare permanentemente il suo contenuto in memoria di massa:

Apertura di un File

FILE *fopen(char *_name_, char *_mode_);

dove:

- ***name*** e` un array di caratteri che rappresenta il nome (assoluto o relativo) del file nel file system
- ***mode*** esprime la modalita` di accesso scelta.
  - **"r"**, in lettura (read)
  - **"w"**, in scrittura (write)
  - **"a"**, scrittura, aggiunta in fondo (append)
  - **"b"**, a fianco ad una delle precedenti, indica che il file e` binario
  - **"t"**, a fianco ad una delle precedenti, indica che il file e`di testo

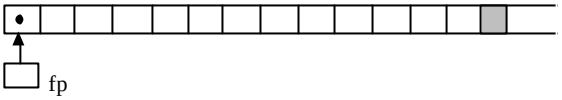
- **Se eseguita con** successo**, l'operazione di apertura ritorna come risultato un** *puntatore* **al file aperto**

- **Se, invece, l'apertura fallisce (ad esempio, perche` il file non esiste), fopen restituisce il valore NULL.**

Apertura in lettura

fp = fopen("filename", "r")

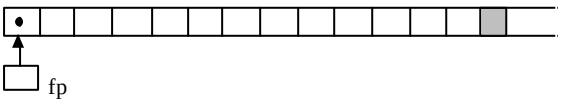
Se il file non e` vuoto:



Apertura in scrittura

fp = fopen("filename", "w")

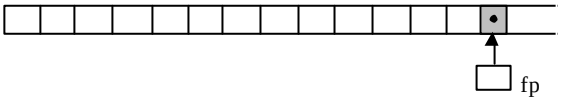
Anche se il file non e` vuoto:



- Se il file esisteva gia`, il suo contenuto viene **perso**.

Apertura in aggiunta (append)

fp = fopen("filename", "a")



Il puntatore al file si posiziona sull'elemento successivo all'ultimo significativo del file ➠ se il file esisteva gia`, il suo contenuto non viene perso.

Ad esempio:

```
File *fp;
fp=fopen("c:\anna\dati", "r");
<uso del file>
```

- fp rappresenta, dall'apertura in poi, il riferimento da utilizzare nelle operazioni di accesso a c:\anna\dati. Esso individua, in particolare:
    - il file
    - l'elemento corrente all'interno del file

Chiusura di un File

Al termine di una sessione di accesso al file, esso deve essere **chiuso**.

L'operazione di chiusura si realizza con la funzione **fclose**:

int fclose (FILE *fp);

dove:
- fp rappresenta il puntatore al file da chiudere.

fclose ritorna come risultato un intero:

- Se l'operazione di chiusura e` eseguita correttamente restituisce il valore 0
- se la chiusura non e` andata a buon fine, ritorna la costante EOF.

Esempio:

```
#include <stdio.h>
void main()
{
  FILE *fp;
  fp = fopen("prova.dat", "w")
  …
   /* scrittura di prova.dat */
  …
  fclose(fp);
  return 0;
}
```


Funzione feof()

Durante la fase di accesso ad un file e` possibile verificare la presenza della marca di fine file con la funzione di libreria:

int feof(FILE *fp);

- feof(fp) controlla se e` stata raggiunta la fine del file fp nella operazione di lettura o scrittura **precedente**. Restituisce il valore 0 (falso logico) se non e` stata raggiunta la fine del file, altrimenti un valore diverso da zero (vero logico).

Lettura e Scrittura di file

Una volta aperto un file, su di esso si puo` accedere in lettura e/o scrittura, compatibilmente con quanto specificato in fase di apertura.

Per file di testo sono disponibili funzioni di:

- Lettura/scrittura **con formato**
- Lettura/scrittura di **caratteri**
- Lettura/scrittura di **stringhe di caratteri**

Per file binari si utilizzano funzioni di:

- Lettura/scrittura di **blocchi**

↦ Ogni operazione di lettura (o scrittura) provoca l'avanzamento del puntatore al file al primo record logico (carattere , linea o blocco) non letto (o libero, nel caso di scrittura).

Accesso a file di testo: Lettura/scrittura con formato

Funzioni simili a **scanf** e **printf**, ma con un parametro aggiuntivo rappresentante il puntatore al file **di testo** sul quale si vuole leggere o scrivere:

Lettura con formato:

Si usa la funzione fscanf:

int fscanf **(FILE *fp,** ***stringa-controllo*****,** ***ind-elem*****);**

dove:

- fp e` il puntatore al file
- *stringa-controllo* indica il formato dei dati da leggere
- *ind-elem* e` la lista degli indirizzi delle variabili a cui assegnare i valori letti.

Esempio:

```
FILE *fp;
int A; char B; float C;
fp=fopen("dati.txt", "r");
fscanf(fp, "%d%c%f", &A, &B, &C);
...
fclose(fp);
```

Restituisce il numero di elementi letti, oppure un valore negativo in caso di errore.

Scrittura con formato:

Si usa la funzione fprintf::

int fprintf **(FILE *fp,** ***stringa-controllo*****,** ***elementi*****);**

dove:

- fp e` il puntatore al file
- *stringa-controllo* indica il formato dei dati da scrivere
- *elementi* e` la lista dei valori (espressioni) da scrivere.

Esempio:

```
FILE *fp;
float c=0.27;
fp=fopen("risultati.txt", "w");
fprintf(fp,"Risultato: %f", c*3.14);
...
fclose(fp);
```

Restituisce il numero di elementi scritti, oppure un valore negativo in caso di errore.

Esempio:

Visualizzazione del contenuto di un file di testo:

```
#include <stdio.h>
void main(){
char buf[80]
FILE *fp;

fp=fopen("testo.txt", "r");
fscanf(fp,"%s",buf);
while(!feof(fp))
{   printf("%s", buf);
    fscanf(fp,"%s",buf);
}
fclose(fp);
}
```

oppure:

```
#include <stdio.h>
void main(){
char buf[80]
FILE *fp;

fp=fopen("testo.txt", "r");
while (fscanf(fp,"%s",buf)>0)
    printf("%s", buf);
fclose(fp);
```

Accesso a file binari: Lettura/scrittura di blocchi

Si puo` leggere o scrivere da un file binario un intero blocco di dati (binari).

Un file binario memorizza dati di qualunque tipo, in particolare dati che non sono caratteri (interi, reali, vettori o strutture).

Per la lettura/scrittura a blocchi e` necessario che il file sia stato aperto in modo ***binario*** (modo "b").

Lettura:

int fread (void **vet*, int *size*, int *n*, FILE **fp*);

Legge (al piu`) *n* oggetti dal file puntato da *fp*, collocandoli nel vettore *vet*, ciascuno di dimensione *size*. Restituisce un intero che rappresenta il numero di oggetti effettivamente letti.

Scrittura

int fwrite (void **vet*, int *size*, int *n*, FILE **fp*);

Scrive sul file puntato da *fp*, prelevandoli dal vettore *vet*, *n* oggetti, ciascuno di dimensione *size*. Restituisce un intero che rappresenta il numero di oggetti effettivamente scritti (inferiore ad *n* solo in caso di errore, o fine del file).

Esempio:

Programma che scrive una sequenza di record (dati da input) in un file binario:

```
#include<stdio.h>
typedef struct{ char nome[20];
                char cognome[20];
                int reddito;
               }persona;

void main()
{ FILE *fp;
  persona p;
  int fine=0;

  fp=fopen("archivio.dat","wb");
  do
  { printf("Dati persona?");
    scanf("%s%s%d ",p.nome,
      p.cognome,&p.reddito);
    fwrite(&p,sizeof(persona),1,fp);
    printf("Fine (si=1,no=0)?");
    scanf("%d", &fine);
  }while(!fine);
  fclose(fp);
}
```


Esempio:

Programma che legge e stampa il contenuto di un file binario:

```
#include<stdio.h>
typedef struct{ char nome[20];
                char cognome[20];
                int reddito;
                }persona;

void main()
{ FILE *fp;
  persona p;

  fp=fopen("archivio.dat","rb");
  fread(&p, sizeof(persona),1, fp);
  while (!feof(fp))
  { printf("%s%s%d",p.nome,p.cognome,
           p.reddito);
    fread(&p,sizeof(persona),1,fp);
  }
  fclose(fp);
}
```


Esempio:

Programma che richiede in input il nome di un file e scrive in questo file un vettore di interi.

```
#include <stdio.h>
#define N 5

void main()
{
FILE *fp;
int i, tab[N]={1,2,3,4,5};
char fname[20];

printf("Inserire il nome del file: ");
scanf("%s",fname);
if ((fp=fopen(fname,"wb"))==NULL)
     printf("Impossibile aprire file
             di uscita\n");
else{
    for(i=0;i<N;i++)
     fwrite(&tab[i], sizeof(int), 1, fp);
    fclose(fp);
}

}
```

Esempio:

Programma che richiede in input il nome di un file di interi e memorizza il contenuto di questo file in un vettore di interi (di al più 40 elementi).

```
#include <stdio.h>
#define MAX 40

void main()
{
 FILE *fp;
 char fname[20];
 int i, j, tab[MAX];

 /* acquisizione del nome del file in fname */
 if ((fp=fopen(fname, "rb"))==NULL)
        printf("Impossibile aprire file
            di ingresso\n");
 else{
    i=0;
    fread(&tab[i++], sizeof(int), 1, fp);
    while(!feof(fp) && i<MAX)
        fread(&tab[i++], sizeof(int), 1, fp);
    fclose(fp);
    /* il valore di i viene incrementato
    anche per la marca di fine file */
    i--;
    for(j=0;j<i;j++)
     printf("%d ", tab[j]);
    }

}
```


Esempio:
Scrittura di record da vettore a file (nome fornito in input).

```
#include <stdio.h>
#define DIM 5
typedef struct  {char nome[15],
                 cognome[15],
                 via[10];
                 int eta;} Persona;
Persona P[DIM];

void main()
{
int i, n;
FILE *fp;
char fname[20];

/* fase di acquisizione di n record */
/* da input memorizzandoli nel vettore P */
/* fname contiene il nome del file di uscita */

if ((fp=fopen(fname, "wb"))==NULL)
        printf("Impossibile aprire il file\n");
else{
       fwrite(P,sizeof(Persona),n,fp);
       fclose(fp);
    }
}
```