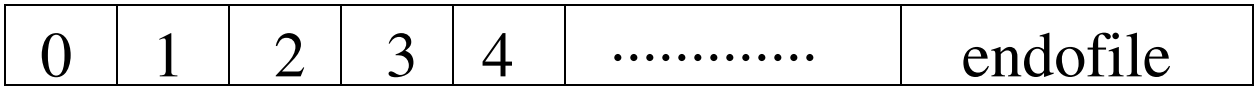
Files in C

Il C vede i file semplicemente come un **flusso** (stream) sequenziale di bytes terminati da un marcatore speciale che determina la fine del file (end-of-file).

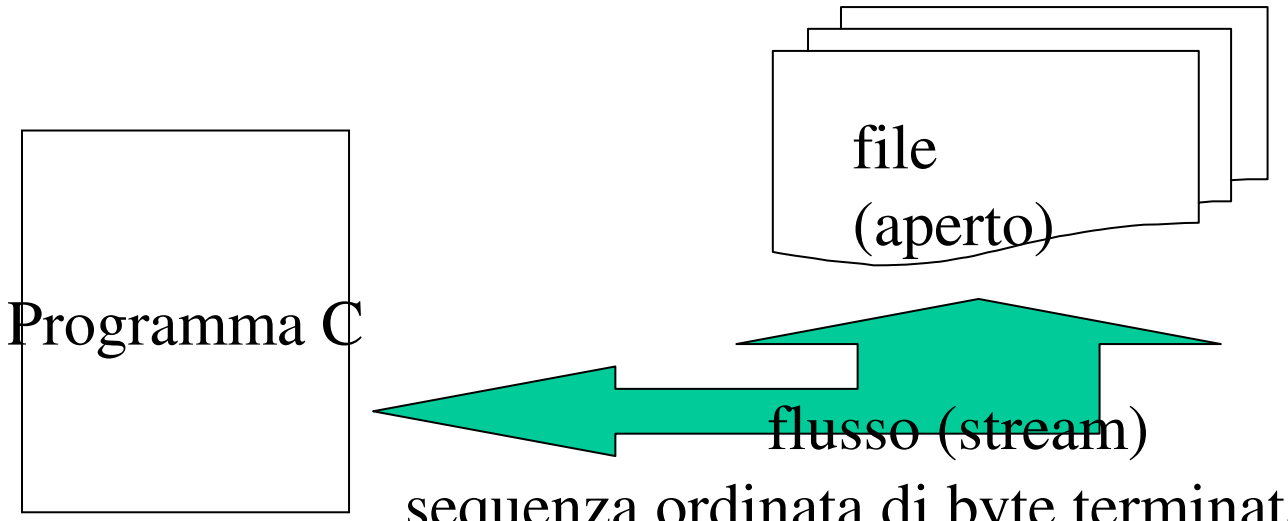


A differenza di altri linguaggi il C **non impone** una **struttura** ai file. Quindi , nozioni come **record di un file** non appartengono al C. Il programmatore dovrà fornire le strutture di file per soddisfare le particolari esigenze di una applicazione.

Files in C

Perché un programma possa lavorare con un file dobbiamo attivare un flusso di dati. Dobbiamo quindi **Aprire** il canale di comunicazione. Diremo quindi **aprire un file**.

Il flusso deve avere una direzione.Diremo che il file è **aperto in lettura** o in **scrittura**

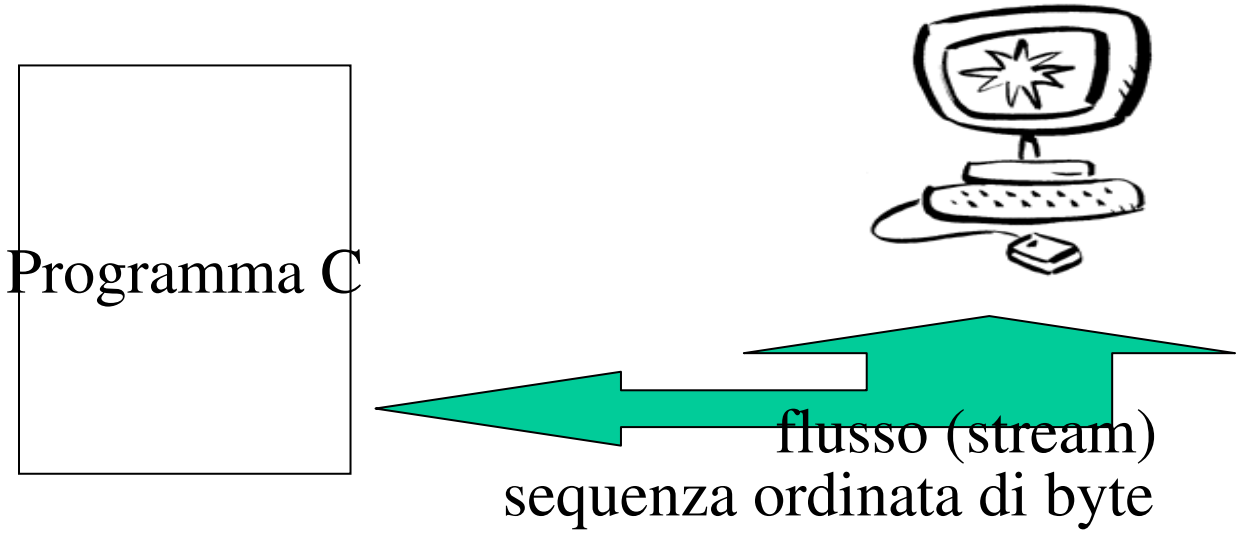


sequenza ordinata di byte terminata da EOF

Esempi di file e loro stream: **stdin, stout**

Standard input:
Consente a un programma di leggere i dati da tastiera

Standard output:
Consente a un programma di scrivere dati sullo schermo



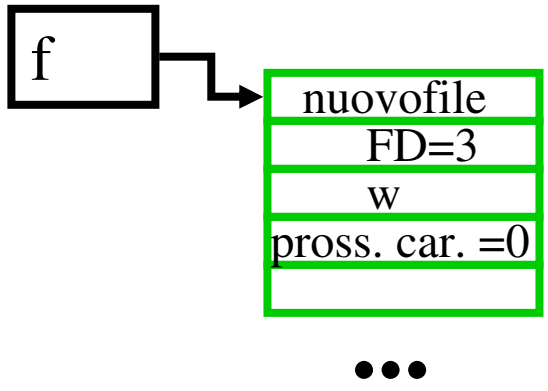
Struttura FILE

FILE *f;

FILE è una struttura definita in stdio.h i cui campi contengono le seguenti informazioni

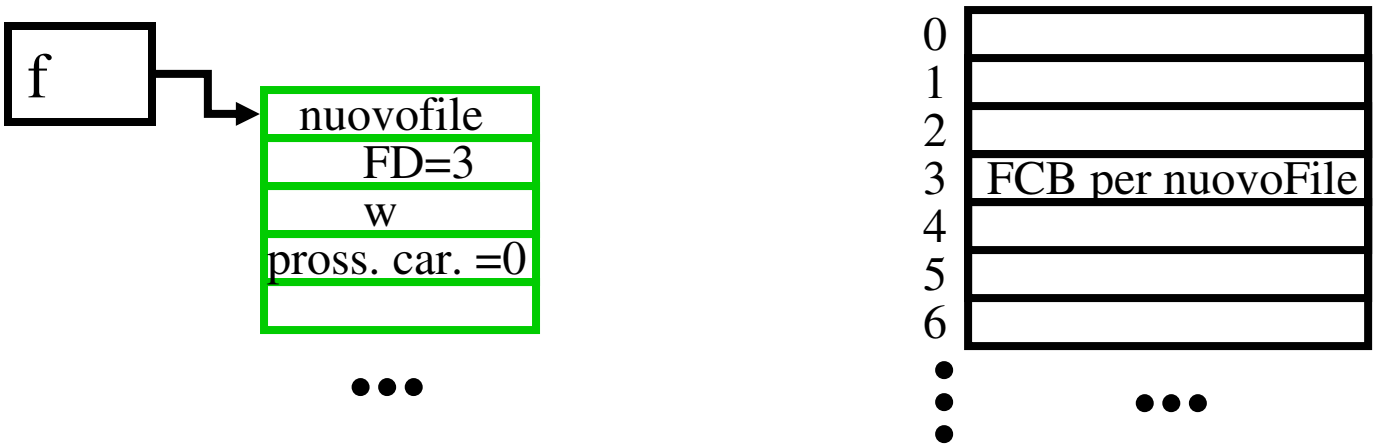
- il nome del file
- File Descriptor
- modalità di accesso (lettura (‘r’), scrittura(‘w’), ...)
- puntatore al prossimo carattere nel flusso

f = fopen(“nuovofile”,”w”)



Descrittore di File

Il descrittore di File è l'indice di un vettore del sistema operativo chiamata tabella dei file aperti (Open File table)



Ogni elemento del vettore contiene un blocco di controllo del file (File Control Block) che il sistema operativo utilizza per amministrare un particolare file

Operazioni di Base su File

- Apertura di un file (apertura del flusso)
- Chiusura di un file (chiusura del flusso)
- Riconoscimento di fine del file

fopen

Prototipo in <stdio.h>

```
FILE *fopen(const char *nomeFile, const char *modo);
```

Parametri

nomeFile	è il nome del file che vogliamo aprire, fopen associa a questo nome un flusso
modo	è la modalità di apertura del file

Risultato fopen() restituisce un puntatore al file aperto, se l'apertura ha avuto successo, NULL altrimenti

Modalità apertura file

r: apre un file di testo per la lettura
w: crea un file di testo per la scrittura o tronca la lunghezza a zero di un file preesistente
a: accodamento, apre o crea un file di testo per scrivere alla fine dello stesso

In tutti i modi di apertura, eccetto quelli "a", l'indicatore di posizione del file è posto all'inizio del file.

fclose

Prototipo in <stdio.h>

int fclose(FILE *stream)

Parametri
stream è il nome del puntatore al file che vogliamo chiudere

Risultato fclose() chiude un file aperto, scaricando nel file tutti I dati presenti nel buffer. Dopo la sua chiamata stream non è più disponibile

feof

Prototipo in <stdio.h>

int feof(FILE *stream);

Parametri
stream è il nome del puntatore al file per il quale vogliamo effettuare il controllo

Risultato feof() restituisce un valore non nullo se stream è all'end-of-file, e 0 altrimenti

Nota feof() verifica se si è raggiunta la fine del file, non modifica lo stato dell'end-of-file.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{FILE *f;
char filename[80];
/* chiede all'utente di specificare un nome di file da aprire*/

printf("Inserisci il nome del file da aprire \n");
gets(filename);
  /*apre il file in lettura */
 f = fopen(filename, "r"))
 if ((f == NULL)
{printf("Il file %s non può essere aperto.\n", filename);}
  /*legge il testo fino alla fine del file*/
while (!feof(f))
     . . . . .
     . . . . .
fclose(f);
 return 0;}
```

Restrizioni

Non é possibile mantenere un file aperto in due modalità differenti.
Esempio. Prima di poter leggere da un file che stiamo creando dobbiamo chiudere il file e aprirlo in lettura.
Comunque esistono modalità di apertura che consentono le due operazioni allo stesso tempo
Esempio r+: apre un file in aggiornamento (lettura e scrittura)

Non é possibile leggere o scrivere in un file aperto in aggiornamento dopo un accesso in scrittura o lettura, a meno di inserire tra le due operazioni una chiamata a fseek(), rewind() o fflush(), operazioni di riposizionamento dell'indicatore di posizione.

rewind

Prototipo in <stdio.h>

void rewind(FILE *stream)

Parametri
stream è il nome del puntatore al file

Risultato La funzione rewind imposta l'indicatore di posizione del file
Per lo stream puntato da stream all inizio del file.

Lettura e scrittura di dati su file di testo

Diversi gradi di granularità:

- carattere
- linea
- blocco

lettura e scrittura per caratteri

- getc() é una macro che legge un carattere da un flusso
- fgetc   come getc(), ma realizzata come funzione
- putc() é una macro che scrive un carattere da un flusso
- fputc() come putc(), ma realizzata come funzione

La macro getc()

Prototipo in stdio.h

int getc(FILE *stream);

Parametri:

- stream é di tipo puntatore a FILE

Risultato:
getc() restituisce il prossimo carattere letto in stream o restituisce EOF se la fine del file é stata raggiunta o si é verificato un errore in lettura, inoltre fa avanzare l'indicatore di posizione.

Nota:
getc() é implementata come una macro.
Come tutte le macro é eseguita più velocemente, ma é suscettibile di effetti collaterali non voluti.

La macro putc

Prototipo in stdio.h:

int putc(int c, FILE *stream);

Parametri:

- c di tipo int é il carattere da scrivere nel file (viene convertito automaticamente nel tipo char prima di essere scritto nel file)

- stream di tipo FILE*, punta al file su cui scrivere

Risultato:
Restituisce il carattere scritto, facendo avanzare l'indicatore di posizione, se la scrittura ha successo, EOF se fallisce

Esempio di uso di getc

```
int copyFileCar(char* nFileIn,char *nFileOut)
{FILE * f1Ptr, *f2Ptr;
char c;

if ((f1Ptr = fopen(nFileIn,"r")) == NULL)
{fprintf(stderr, "il file %s non può essere aperto, in lettura \n
",nFileIn);
 return FALLIM;
}
if (( f2Ptr = fopen(nFileOut,"w")) == NULL)
 {fclose(f1Ptr);
 fprintf(stderr, "il file %s non può essere aperto, in scrittura \n
",nFileOut);
 return FALLIM;
}
while (!feof(f1Ptr) )  putc(getc(f1Ptr),f2Ptr);
          /* alternativa :
          while ((c=getc(f1Ptr)) != EOF)
          putc(c,f2Ptr); */
fclose(f1Ptr);
fclose(f2Ptr);
return SUCCESSO;
}
```

La funzione fgetc

fgetc() lavora esattamente come getc() eccetto che si tratta di una funzione.

Prototipo in stdio.h

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

Parametri:

stream é di tipo puntatore a FILE

Risultato:

fgetc() restituisce, come un intero, il prossimo carattere letto in stream o restituisce EOF se la fine del file é stata raggiunta o si é verificato un errore in lettura, inoltre fa avanzare l'indicatore di posizione.

La funzione fputc

fputc() lavora esattamente come putc() eccetto che si tratta di una funzione.

Prototipo in stdio.h:

```
int fputc(int c, FILE *stream);
```

Parametri:

- c — di tipo int é il carattere da scrivere nel file (viene convertito automaticamente nel tipo char prima di essere scritto nel file)
- stream — di tipo FILE*, punta al file su cui scrivere

Risultato:
Restituisce il carattere scritto, facendo avanzare l'indicatore di posizione, se la scrittura ha successo, EOF se fallisce

LETTURA E SCRITTURA PER LINEE

- fgets() legge una linea da un file
- fputs() scrive una linea in un file

La funzione **fgets**

Prototipo in stdio.h:

```
char *fgets(char *s, int n, FILE *stream);
```

Parametri:

s di tipo char * è la stringa che conterrà la linea letta

n di tipo int è il massimo numero di caratteri letti, compreso il carattere nullo!

stream di tipo FILE * punta al file da cui leggere

Risultato:

fgets() legge i caratteri in sequenza stream cominciando dalla corrente posizione, li immagazzina in s e restituisce un puntatore a s, se la lettura ha avuto successo, aggiungendo anche il newline e il carattere di terminazione di stringa \0 . La lettura si ferma se viene letto un **newline**, **EOF** o dopo aver letto n caratteri Se raggiunge l'EOF prima di leggere alcun carattere s resta inalterato e fgets() restituisce un puntatore nullo (NULL). Se occorre un errore fgets() restituisce un puntatore nullo e il contenuto di s può essere corrotto.

fputs

Scrive un vettore di caratteri in un file

Prototipo in stdio.h:

int fputs(const char *s, FILE *stream);

Parametri:

- s di tipo char * è la stringa da scrivere nel files
- stream di tipo FILE * punta al file da cui leggere

Nota:

La fputs() scrive l'array puntato da s nel file e avanza l'indicatore di posizione del file A differenza di puts() non aggiunge il newline ('\n'), che può non essere alla fine di *s*.

Valore restituito

fputs() restituisce uno zero se ha successo e un valore non nullo altrimenti.

Esempio di uso di fgets e fputs

```
int copyFileLine (char* nFileIn,char *nFileOut)
{FILE * f1Ptr, *f2Ptr;
char line[LINESIZE];
if ((f1Ptr = fopen(nFileIn,"r")) == NULL)
{fprintf(stderr, "il file %s non può essere aperto, in lettura \n ",nFileIn)
 return FALLIM;}
if (( f2Ptr = fopen(nFileOut,"w")) == NULL)
 {fclose(f1Ptr);
 fprintf(stderr, "il file %s non può essere aperto, in scrittura \n ",nFileO
 return FALLIM;}
 if (line != NULL)
while (fgets(line,LINESIZE-1,f1Ptr) !=NULL)
 {fputs(line,f2Ptr);
 fputc('\n',f2Ptr);}
fclose(f1Ptr);
 fclose(f2Ptr);
 return SUCCESSO;}
```

Lettura e scrittura di blocchi

Come per stdin e stdout si usano le funzioni
printf e scanf per scrivere in output e leggere
dall'input dati con un formato,

per lettura e scrittura su file abbiamo le funzioni
fscanf
fprintf

fprintf

Prototipo in stdio.h:

int fprintf(FILE *stream, const char *formato,.....);

Parametri:

- formato stringa che specifica in che modo devono essere convertiti per
  L'output gli argomenti specificati in .....

- stream di tipo FILE*, punta al file su cui scrivere
- ..... argomenti

Risultato:
La funzione fprintf invia il proprio output nello stream puntato da stream, in modo controllato dalla stringa formato, che specificherà in che modo debbano essere convertiti per l'output gli argomenti successivi. Qualora non ci siano argomenti sufficienti il comportamento sarà indefinito.funzione fprints restituirà il numero di caratteri inviati in output o un valore negativo qualora si verifichi un errore

fscanf

Prototipo in stdio.h:

int fscanf(FILE *stream, const char *formato,.....);

Parametri:

- **formato** stringa che specifica le sequenza di input ammissibili e come devono essere convertite

- **stream** di tipo FILE*, punta al file da cui leggere
- ..... argomenti

Risultato:
La funzione **fscanf** legge l'input dallo stream puntato da **stream**, in modo controllato dalla stringa **formato**, che specificherà il modo in cui queste dovranno essere convertite per l'assegnamento, utilizzando gli argomenti successivi come puntatori agli oggetti che riceveranno l'input convertito sequenze di input ammissibili. La fscanf restituisce il valore della macro EOF se si verifica un errore. Altrimenti restituisce il numero di elementi assegnati, che potranno essere anche inferiori a quelli specificati o 0 in caso di un prematura fallimento di corrispondenza

Un esempio

Scriveremo un programma per creare un file ad accesso sequenziale che potrebbe essere usato in un sistema di contabilità del credito per aiutare a tenere il computo delle somme dovuta dai clienti debitori di un'azienda.

Per ogni cliente il programma deve trattare

Il numero di conto,
Il nome del cliente
La quantità dovuta

I dati ottenuti per ogni cliente cosituiscono il "record" dedicato ad ognuno di loro

Un esempio: creare un file ad accesso sequenziale

```
main(){
int num_conto;
char nome[30];
float saldo;
FILE   *cfPtr /* puntatore al file clienti.dat */

if ((cfPtr = fopen(clienti.dat",”w")) == NULL)
  printf(“Il file non può essere aperto")
else{
printf(“Immetti il conto, il nome, eil saldo\n");
printf(“Immetti EOF per terminare l’immissione\n");
printf(“>");
scanf(“%d%s%f",&num_conto,nome,&saldo);
while(!foeof(stdin)){
            fprintf(cfPtr, “%d%s%f"\n,num_conto,nome,saldo),
            printf(“>");
            scanf(“%d%s%f",&num_conto,nome,&saldo);
}
fclose(cfPtr);
}
return 0;
}
```

Esecuzione

Immetti il conto, il nome, e il saldo
Immetti EOF per terminare l'immissione;

> 100 Bianchi 100.98
> 200 Rossi 200.34
> 400 Dotti 123
> CTRL Z

Un esempio: leggere e stampare un file sequenziale

```
main(){
int num_conto;
char nome[30];
float saldo;
FILE   *cfPtr /* puntatore al file clienti.dat */

if ((cfPtr = fopen(clienti.dat",”w")) == NULL)
  printf(“Il file non può essere aperto")
else{
printf("%-10s%-13s%", “Num Conto",”Nome",”Saldo");
fscanf(cfPtr,"%d%s%f",&num_conto,nome,&saldo);

while(!foeof(cfPtr)){
            printf("%-10s%-13s%", num_conto,nome,saldo);
            fscanf(cfPtr,"%d%s%f",&num_conto,nome,&saldo);
}
fclose(cfPtr);
}
return 0;
}
```

Output

Num Conto	Nome	Saldo
100	Bianchi	100.98
200	Rossi	200.34
400	Dotti	123