

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

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

Gentile sig. Rossi,
con la presente le comunico che
è scaduto il termine per il pagamento
dell'Assicurazione della sua automobile.

⇒ Il record logico puo` essere il singolo carattere, la parola, oppure la linea.

Il **file di testo** è un file che contiene caratteri ASCII

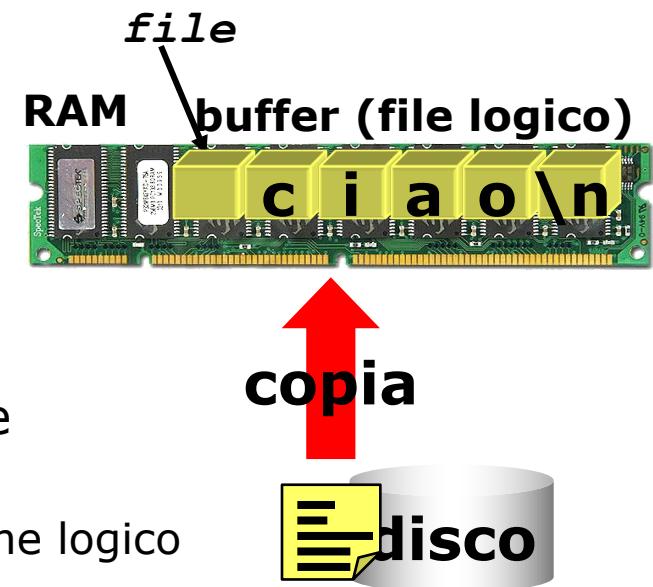
Il **file fisico** è immagazzinato su una memoria secondaria (disco)

Può essere manipolato con le funzioni che il sistema operativo mette a disposizione dei programmi

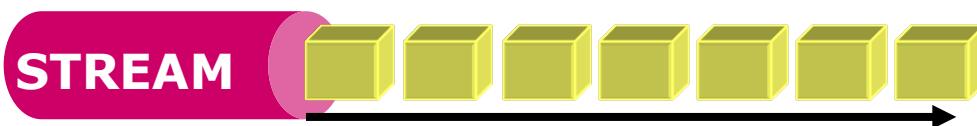
Nome del file fisico: una stringa che indica il percorso e il nome del file sul supporto di memoria

Il **file logico** (buffer) è una copia residente in RAM di dati contenuti in un file fisico

Nel programma viene identificato da un nome logico



Uno **stream** è un'astrazione, che rappresenta un "qualcosa" da o verso cui "fluisce" una sequenza di bytes



Esempi di stream:

Input da tastiera, output a schermo

File in lettura, file in scrittura

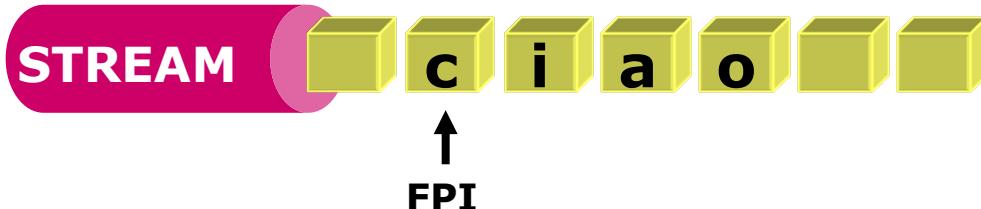
Buffer

Canali di trasmissione di rete

Un concetto importante è la posizione corrente in uno stream (*file position indicator*), che coincide con l'indice del prossimo byte che deve essere letto o scritto

Ogni operazione di I/O modifica la posizione corrente

La posizione corrente può essere ricavata o modificata direttamente usando particolari funzioni



Un file di testo può essere immaginato come una successione di caratteri leggibili, separati da carattere speciali:

La **posizione corrente nel file (file position indicator)**, durante la lettura o la scrittura dei caratteri è memorizzata nell'oggetto file FPI può essere spostato in qualsiasi parte del file con l'uso di opportuni metodi

Spazi (blank, tab,...)

Terminatore di linea
(end-of-line)

Corso di Programmazione I<eol>
Gruppo 3<eol>
Prof. Silvia Rossi<eol><eof>

FPI

Terminatore di file
(end-of-file)

Supponiamo che il contenuto di un file di nome fisico *prova.txt*, che si trova sul disco rigido sia:

Nel mezzo del cammin di nostra vita<eoln>
mi ritrovai in una selva oscura<eoln><eof>

Per scrivere questo testo è sufficiente utilizzare un qualsiasi editor (come lo stesso sistema di sviluppo DevCpp oppure Blocco Note sotto Windows).

Sistema di computer	Combinazione di tasti
UNIX	<return> <ctrl> d
Windows	<ctrl> z
Macintosh	<ctrl> d

Per gestire un file in un programma C occorre:

1. Includere `<stdio.h>`
2. dichiarare una variabile di tipo FILE
3. aprire il file creando una corrispondenza tra nome fisico e nome logico del file
4. leggere o scrivere i dati nel file
5. CHIUDERE IL FILE

Dichiarazione *FILE *cfPtr*

Indica l'elemento corrente all'interno della sequenza

L'apertura del file logico con il conseguente collegamento al file fisico è svolta dalla seguente procedura:

```
cfPtr = fopen("nomefile", "w")
```

- Se il file non esiste viene creato da fopen
- Se il file esiste i suoi contenuti saranno eliminati



Una volta creata la corrispondenza tra file fisico e file logico, il sistema operativo si occuperà di travasare i dati dall'uno all'altro.

Quando ordiniamo di scrivere sul file logico:

- i dati inviati vengono memorizzati temporaneamente nel buffer.
- Nel momento in cui tale zona di memoria viene riempita oppure il file viene chiuso, allora il computer provvederà ad inviare tutti i dati al file fisico.

Nella fase di lettura:

- il computer legge dal file fisico una quantità di dati tale da riempire il buffer (oppure legge tutti i dati del file se questi sono in quantità inferiore alla lunghezza del buffer);
- ogni istruzione di lettura successiva continuerà a leggere dati dal buffer finché esso non si svuota;
- a quel punto nuovi dati vengono prelevati dal file ed inseriti nel buffer per essere letti successivamente.

La **chiusura del file logico** con conseguente rilascio del collegamento col file fisico è svolta dal seguente metodo (procedura):

```
fclose(cfPtr);
```

Alla chiusura del file logico le modifiche in esso registrate sono definitivamente scritte sul file fisico

L'apertura di un file (in lettura e/o scrittura) potrebbe non aver successo:

- Si apre un file in una modalità protetta (scrittura, lettura, entrambi)
- Si apre un file senza averne i permessi opportuni
- Si apre un file indicando un nome (o percorso) errato
- ...

Esempio:

```
if( (cfPtr = fopen("nomefile", "w")) ==NULL) {  
    printf("File could not be opened");  
}
```

Per leggere da un file:

```
fopen("clients.dat","r");
```

Altre modalità:

“a” append

“r+” file già esistente in lettura e scrittura

“w+” file nuovo in lettura e scrittura

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

File di testo: sono disponibili funzioni di:

- Lettura/scrittura con formato: fscanf, fprintf
- Lettura/scrittura di caratteri: fgetc, fputc
- Lettura/scrittura di stringhe di caratteri: fgets, fputs.

```
int account;
char name [30];
double balance;

fscanf( cfPtr, "%d%s%lf", &account, name, &balance);
```

Per determinare la fine di un file si usa il metodo (funzione)

```
feof( cfPtr )
```

ritorna `true` se il *FPI* è sul carattere `<eof>`, `false` altrimenti:

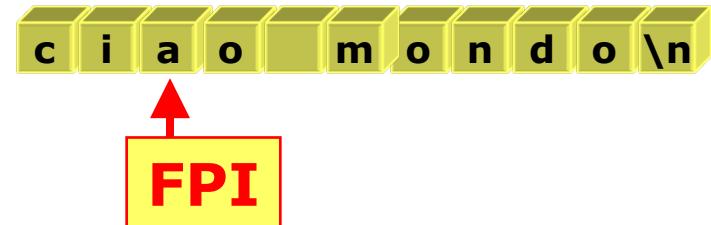
vale `true` se non ci sono più caratteri da leggere o se si è scritto un carattere alla fine del file

Esempio:

```
while (!feof( cfPtr )) {  
    fscanf( cfPtr, "%d%s%lf", &account, name, &balance);  
    printf( "%d %s% f \n", account, name, balance);  
}
```

Per leggere un carattere alla volta in un file si usa il metodo:

```
c = fgetc(cfPtr)
```



c è la variabile di tipo **char** che conterrà il carattere letto dal buffer

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

Trasferisce nella stringa *s* i caratteri letti dal file puntato da *fp*, fino a quando ha letto *n*-1 caratteri, oppure ha incontrato un newline, oppure la fine del file.

La fgets mantiene il newline nella stringa *s*.

Restituisce la stringa letta in caso di corretta terminazione;
\0' in caso di errore o fine del file.

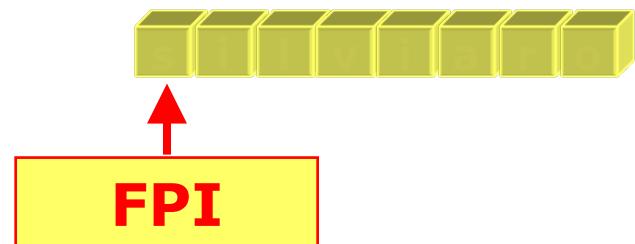
```
rewind( cfPtr );
```

```
fprintf(cfPtr, "%d %s %.2f\n", account, name, balance)
```

Per scrivere un carattere alla volta in un file si usa le funzione:

```
fputc('c', file)
```

c è la variabile **char** che contiene
il carattere da scrivere nel file



```
int *fputs (char *s, FILE *fp);
```

Trasferisce la stringa s (terminata da '\0') nel file puntato da fp.
Non copia il carattere terminatore '\0' ne` aggiunge un newline finale.

Restituisce l'ultimo carattere scritto in caso di terminazione corretta; EOF altrimenti.

Le operazioni standard iniziano con un file vuoto

Se il file esiste → il contenuto viene perso

append:

Se il file non esiste → viene creato

Se il file esiste → I dati vengono aggiunti alla fine del file

Esempio: lettura di un archivio studenti

Si abbia un file di tipo testo, "prova01.txt", contenente le seguenti informazioni rispetto agli studenti di una certa università:

- cognome e nome dello studente seguito dal
 - numero di matricola,
 - 4 numeri interi che rappresentano il voto dei primi 4 esami (lo zero indica che l'esame non è stato superato)

Scrivere un programma che, assegnata una media M in input, stampi il cognome, nome e numero di matricola di tutti gli studenti che hanno una media maggiore o uguale ad M

Esempio:

Input: M=26 Output: Caio Mario 15340
Ferrara Ciro 13124

prova01.txt

deRossi Mario 13240 28 25 0 24<eol>
Caio Mario 15340 28 0 26 24<eol>
Bianchi Antonio 12540 26 20 30 25<eol>
Ferrara Ciro 13124 30 28 0 27<eol><eof>

Esempio: lettura di un archivio studenti

Algoritmo

```
leggi(media)
apri(file)
while (not fine(file)):
    leggi_riga_finoa(file, nome, ';')
    leggi_parola(file, matricola)
    somma ← 0
    nvoti ← 0
    for i from 1 to 4:
        leggi_parola(file, voto)
        if voto > 0:
            somma ← somma + voto
            nvoti ← nvoti + 1
    if (somma > 0) and (somma/nvoti ≥ media):
        stampa(nome, matricola)
chiudi(file)
```

prova2.txt

de Rossi Mario	13240	28	25	0	24	<eol>
Caio Mario	15340	28	0	26	24	<eol>
Bianchi Antonio	12540	26	20	30	25	<eol>
Ferrara Ciro	13124	30	28	0	27	<eol><eof>

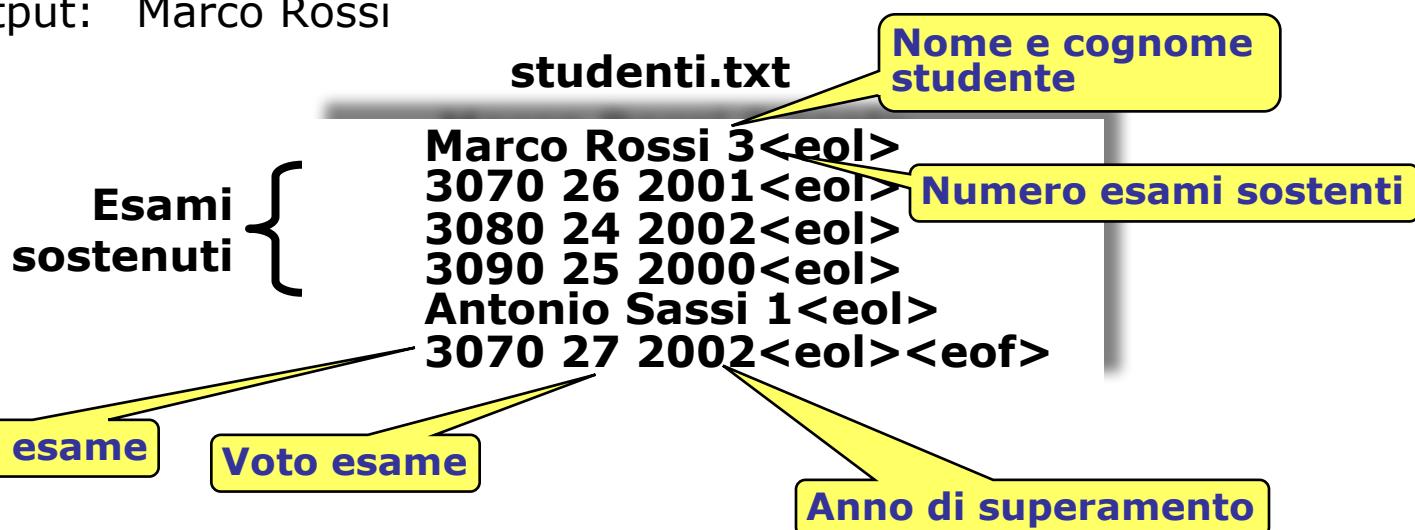
Esercizi:

1. Sia dato un file testo di nome fisico "studenti.txt" contenente informazioni sugli studenti di un corso (strutturate secondo l'esempio in figura). Dati un intero C rappresentante un codice d'esame, un intero M rappresentante un voto ed un intero N rappresentante un anno, scrivere un algoritmo che dia in output il nome degli studenti che hanno sostenuto l'esame di codice C prima dell'anno N con un voto $\geq M$.

Esempio:

Input: 3070 24 2002

Output: Marco Rossi



2. Un file testo "parole.txt" contiene, per ogni riga, una serie di parole, ognuna delle quali è seguita da uno spazio o dal carattere <eol>. Ogni parola è costituita da caratteri alfabetici e può essere composta da sole lettere maiuscole, sole lettere minuscole o da entrambe. Scrivere una procedura che legga il file "parole.txt" e costruisca due file:

nel primo file andranno inserite, una per riga, le parole contenenti solo lettere maiuscole;

nel secondo andranno inserite, sempre una per riga, le parole contenenti solo lettere minuscole.

Le parole contenenti sia lettere maiuscole che lettere minuscole non andranno memorizzate

Esempio:

parole.txt

**CASA libro DADO trENO <eoln>
mattone penna AUTO <eoln>
ROSA acqua raDIO <eoln><eof>**



**CASA<eol>
DADO<eol>
AUTO<eol>
ROSA<eol><eof>**

**libro<eoln>
mattone<eoln>
penna<eoln>
acqua<eoln><eof>**

3. Sia dato un file testo "parole.txt" composto da sole lettere maiuscole. Ogni linea di testo è costituita da un carattere seguito da uno spazio e da una serie di parole tutte separate da uno spazio (compreso l'ultima). Tutte le parole di un rigo dovrebbero iniziare o terminare con il carattere di inizio riga.

Scrivere una procedura che legga il file "parole.txt" e stampi a video tutte le parole che non verificano la condizione suddetta

Esempio:

parole.txt

M MONDO MARE <eol>
O AGO NERO OTTOBRE <eol>
P PANE PERA <eol>
A CASA <eol><eof>



Output:

parole.txt

M MONDO GARE <eol>
O OCA POLLO <eol>
P PANE SERA <eol>
F GATTO <eol><eof>



**Output: GARE
SERÀ
GATTO**

4. Siano dati due file testo: "studenti.txt" e "tesi.txt". Il primo contiene informazioni riguardanti tutti gli studenti di un corso di laurea; il secondo contiene informazioni riguardanti i soli studenti del medesimo corso di laurea che hanno già fatto richiesta della tesi (vedi figura) Scrivere una procedura che mostri a video la matricola e la media di tutti gli studenti che hanno richiesto la tesi e che hanno svolto tutti gli esami

studenti.txt

4010<eol>

Rossi Marco<eol>

18 0<eol>

Nome e cognome

matricola

Num. esami
sostenuti

Num. esami
da sostenere

...
programmazione A^ 26<eol>

4050<eoln>

Antonio Neri<eol>

10 8<eoln>

Voto esame

Nome esame

...

Architettura di elaboratori^ 22<eol>

4060<eoln>

Bianchi Matteo<eol>

16 2<eoln>

...

Analisi matematica^ 23<eol><eof>

tesi.txt

4010<eol>

Rossi Marco<eol>

4060<eol>

Bianchi Matteo<eol><eof>

Entrambi i file sono ordinati per numero di matricola crescente.