

**Fondamenti di Informatica**  
**Ingegneria Clinica**  
**Lezione 24/01/2011**



**Raffaele Nicolussi**  
**FUB - Fondazione Ugo Bordoni**  
**Via del Policlinico, 147 – 00161**  
**Roma**

## I/O in C

- Il sistema di I/O del C è una interfaccia **uniforme** per il programmatore **indipendentemente dal dispositivo di lettura e scrittura utilizzato**
  - Scrivere o leggere da file o da video/tastiera non comporta differenze
  - Il sistema di I/O del C crea un livello di astrazione tra il programmatore e l'oggetto (sia esso un file o dispositivo)

Viene creato un **meccanismo di astrazione** tramite un **flusso di dati o stream**

# Stream

- Uno **stream (flusso di dati)** fornisce un canale di comunicazione tra il programma e l'esterno (file o device)
- Uno **stream** è una struttura costituita da una **sequenza di byte**, in numero teoricamente infinito, terminante con un apposito carattere che ne identifica la fine
- Gli **stream** vengono associati (con opportuni comandi)
  - **ai dispositivi fisici** collegati al computer (tastiera, video, stampante) o
  - **a file** residenti sulla memoria di massa

## La funzione di output printf()

- **Produce una sequenza di caratteri che viene inviata allo *stream stdout* (*video*)**

```
printf("la signora vende %d %s per  
      %f %s",  
      20, "orate", 3.55, "Euro");
```

# La funzione di output printf()

- **Parametri:**

**printf (<stringa di controllo>  
<altri-argomenti>)**

- **Stringa di controllo:** stringa che contiene anche specifiche di conversione
- **Altri-argomenti:** espressioni convertite in base ai formati di conversione specificati nella stringa di controllo

## La funzione di input scanf()

- Legge i caratteri dallo **stream stdin (tastiera)**, ossia lo stream di input e li deposita convertendoli nello specifico formato nelle variabili il cui indirizzo e' specificato
- Es.
  - `char a, b, c; int n; double x;`
  - `scanf (“%c%c%c%d%lf”,  
          &a, &b, &c, &n, &x);`

# La funzione di input scanf()

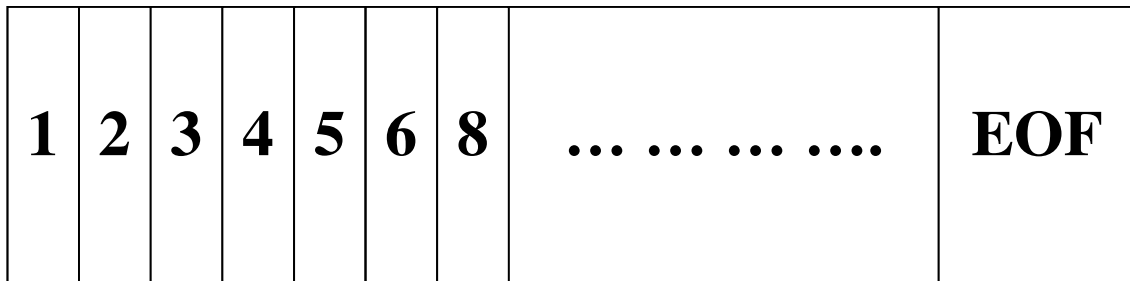
- **Parametri**

**scanf (<stringa di controllo>  
<altri-argomenti>)**

- **Stringa di controllo:** specifica di conversione per la rappresentazione interna
- **Altri-argomenti:** elenco di indirizzi di memoria dove depositare i dati convertiti

# File

- Flusso sequenziale di byte terminato dal carattere di terminazione chiamato **end-of-file (EOF)**.



- Il file deve essere connesso al programma tramite l'astrazione dello **stream**
  - **L'apertura e la chiusura dello stream associato al file non sono automatici come invece nel caso di stdin, stdout, stderr**



## Le funzioni fprintf() e fscanf()

- Sono le versioni di **printf** e **scanf** relative ad un generico file esterno delle
  - **fprintf** (*<descrittoreFile>*  
*<stringa di controllo>*  
*<altri-argomenti>*)
  - **fscanf** (*<descrittoreFile>*  
*<stringa di controllo>*  
*<altri-argomenti>*)

## <descrittoreFile>

- Cosa è **<descrittoreFile>**:
  - Si deve creare una **connessione** tra:
    - **il mondo interno al programma** (tutto in memoria centrale)
  - ed
    - **il mondo esterno** (rappresentato dai file residente nelle memorie di massa)
- È legato al **nome fisico** del file e ad altre caratteristiche strutturali

## <descrittoreFile>

### Per usare file esterni

- Si deve introdurre questo **“rappresentante interno”** del mondo esterno
- Si deve creare una **associazione temporanea tra mondo esterno e “rappresentante interno”**

# Il descrittore di file è una variabile

- Specificare un nome e assegnare un tipo  
**FILE \*<identificatore>;**

Esempio:

```
FILE *ingressoDsc;
```

- La sequenza **FILE \*** si riferisce ad uno specifico tipo **puntatore a file esterno**
- **ingressoDsc** è una variabile di tipo **puntatore a file**
- **Attenzione:** manca ancora il legame con l'esterno

## Connessione

- La funzione **fopen** crea la connessione tra il **descrittore** ed il **file vero e proprio** (esterno)

```
<descrittoreFile>=  
fopen(<nomefileesterno>  
      <modalità di uso>);
```

```
ingressoDsc=  
fopen("filedaleggere.txt",  
      "r");
```

## Modalità di uso

- Si deve specificare se il file va usato in lettura o scrittura
- Si ricordi la distinzione tra *stdin* ed *stdout*
  - File in lettura **r** (read)
  - File in scrittura **w** (write)
  - Scrittura alla fine del file **a** (append)

```
FILE *uscitaDsc;  
uscitaDsc=  
fopen("filedascrivere.txt",  
      "w");
```

## Chiusura della connessione

- **Quando il file esterno non serve più, deve essere chiuso**
- **Questo si ottiene cancellando esplicitamente la connessione con l'esterno**

```
fclose(<descrittoreFile>);
```

## Chiusura della connessione

```
int main (void) {  
    altre dichiarazioni di variabili  
    FILE *inputDsc;  
    FILE *outputDsc;  
    .....  
    inputDsc =  
    fopen(“filedatiingresso”, “r”)  
    outputDsc =  
    fopen(“filedatiuscita.txt”, “w”)  
    .....  
    fclose(inputDsc);  
    fclose(outputDsc);  
}
```



## Eccezioni

- **Nella comunicazione con il mondo esterno può accadere che il file non si apra correttamente.**
- **Questo viene segnalato da un valore particolare del descrittore di file, indicato con **NULL**.**
- **Dopo **fopen( )** inserire sempre un controllo**

***if (inputDsc == NULL)***

.....

## Eccezioni

```
if (inputDsc == NULL)
```

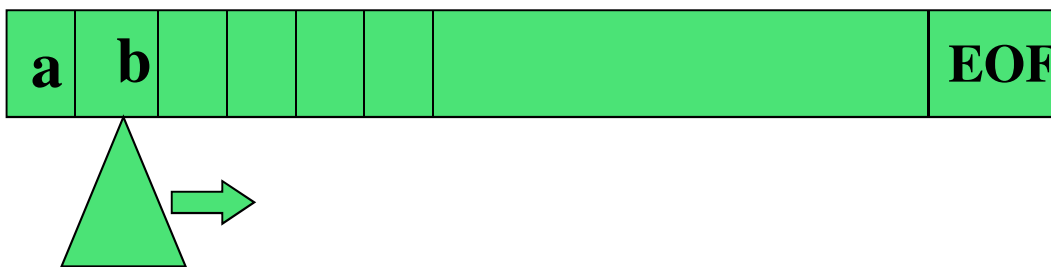
```
    printf (“errore nella apertura del  
        file di ingresso”);
```

```
if (outputDsc == NULL)
```

```
    printf (“errore nella apertura del  
        file di uscita”);
```

## La fine del file (simbolo EOF)

- La fine del file viene segnalata da un carattere particolare il simbolo di End-of-File o **EOF**



- Ci sono vari modi di verificare la terminazione di un file
- Modello semplice per il file:
  - un *nastro* con delle informazioni ed una *testina di lettura* che è il descrittore. La testina viene spostata da istruzioni come **fscanf( )** ed **fprintf( )**

## La fine del file (simbolo EOF)

- La codifica del carattere **EOF** dipende dalla macchina
  - **Ctrl-z** per **Windows**
  - **Ctrl-d** per **Mac**
  - **Ctrl-x** per **Unix e Linux**

## Riassumendo

1. Viene dichiarato il descrittore del file

**FILE \*<descrittore>;**

2. Il file si apre con la funzione **fopen ()**

**<descrittore>=**

**fopen(<nome\_file>,**

**<modo\_apertura>);**

- <nome\_file> è il nome fisico del file, come è chiamato sul disco, es. *ingresso.dat*
- <modo\_apertura>:
  - “w” se il file è aperto in scrittura
  - “r” se il file è aperto in lettura

3. Il file viene chiuso

**fclose(<descrittore>);**

## Riassumendo(2)

```
FILE *fp;
```

```
fp=fopen("test.dat", "r");
```

```
FILE *fptr;
```

```
fptr=fopen("prova.dat", "w");
```

```
fclose(fp);
```

```
fclose(fptr);
```

## Esempio

**/\* Leggere un file di testo e stampare ogni parola in una singola riga, ad esempio:**

**input: ciao a tutti!**

**output:**

**ciao**

**a**

**tutti!**

**\*/**

Esiste già un file che deve essere aperto in lettura e analizzato per produrre il risultato desiderato

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

```
int main (void){  
    char ch;  
    FILE *fd; /*descrittore del file*/  
  
    /* apertura in lettura e verifica */  
    fd = fopen("parole3.txt", "r");  
  
    if!(NULL==fd){  
        /* stampa dati del file (inserire istruz.)*/  
        }  
    else  
        printf ("\nErrore di  
            apertura!");  
  
    fclose(fd);  
    return 0;  
  
}
```



## Esempio: stampa dei caratteri

```
Finché (ci sono caratteri del  
file) do  
{prendi un carattere e  
analizzalo }
```

```
Finché (non arriva la fine del file)  
do  
{  
    leggi un carattere dal file  
    se (è uno spazio)  
        stampa carattere di riga  
nuova  
    altrimenti  
        stampa il carattere stesso  
}
```

```
while (!(EOF==fscanf (fd, "%c", &ch)))
{
    if(ch==' ')
        printf("\n");
    else
        printf("%c", ch);
}
```

**fscanf** torna un valore che è pari ad **EOF** quando legge la fine del file

# Operatori per leggere e scrivere caratteri

- **Da tastiera e video**
  - **getchar( ) (lettura)**
  - **putchar( ) (scrittura)**
- **Da file**
  - **fgetc(<filedescrittore>);**
  - **fputc (<intero>, <filedescrittore>):**

•**Sono molto usate nella manipolazione di caratteri**

•**Si noti che le get( ) ritornano un valore che deve essere assegnato ad una variabile opportuna**

## Altra codifica (2)

```
while(!(EOF==(ch = fgetc(fd))))
{
    if(ch==' ')
        printf("\n");
    else
        printf("%c", ch);
}
```

Questa tecnica è usata anche nell'esempio di Deitel& Deitel (pag.96) per leggere con **getchar** da **stdin** (standard input)

## Ancora una codifica(3)

```
while( (ch = fgetc(fd)) != EOF){
    if(ch==' ')
        putchar('\n');
    else
        putchar(ch);
}
```

Qui abbiamo solo scritto diversamente il test ed usato il `putchar()` invece del `printf()`

## **Esempio di scrittura su file**

- **Memorizzare su un file i seguenti dati:**
  - **Numero conto**
  - **Bilancio**
  - **Per 10 clienti**

```

#include <stdio.h>
int main (void) {
    int conto, i; float bilancio;
    FILE *cfPtr; /* descrittore file */
    if ((cfPtr = fopen("clienti.dat", "w")) ==
        NULL)
        printf("Il file non puo' essere aperto");
    else {
        for (i=1; i<=10; i++) {
            printf("Immetti il conto e il bilancio\n");
            scanf("%d%f", &conto, &bilancio);
            fprintf(cfPtr, "%d %f\n", conto,
                bilancio);
        }
        fclose(cfPtr);
    }
    return 0;
}

```

## **Esempio di lettura da file**

- **Leggere dal file clienti.dat i seguenti dati:**
  - **Numero conto**
  - **Bilancio**



```

#include <stdio.h>
int main (void) {
    int conto, i; float bilancio;
    FILE *cfPtr; /* descrittore file */
    if ((cfPtr = fopen("clienti.dat", "r")) ==
        NULL)
        printf("Il file non puo' essere aperto");
    else {
        printf("Conto \t Bilancio\n");
        for (i=1; i<=10; i++) {
            fscanf(cfPtr, "%d%f", &conto,
                &bilancio);
            printf("%d \t\t %f\n", conto, bilancio);
        }
        fclose(cfPtr);}
return 0;
}

```

## Modalità di accesso ai file

<b>r</b>	Aprire un file esistente in modalità lettura ( <b>read</b> ) posizionandosi all'inizio del file
<b>w</b>	Crea un nuovo file e lo apre in modalità scrittura ( <b>write</b> ) posizionandosi all'inizio del file. Se il file esiste i dati precedenti vengono eliminati.
<b>A</b>	Aprire un file esistente in modalità <b>Append</b> ; la scrittura avviene a partire dalla fine del file

## Modalità di accesso ai file

<b>r+</b>	Aprire un file esistente in modalità lettura e scrittura posizionandosi all'inizio del file
<b>w+</b>	Crea un nuovo file e lo apre in modalità scrittura e lettura. Se il file esiste i dati precedenti vengono eliminati.
<b>a+</b>	Aprire un file esistente o ne crea uno nuovo in modalità <b>Append</b> ; la scrittura avviene a partire dalla fine del file mentre la lettura può avvenire da una posizione qualsiasi.

## Modalità di accesso ai file

<b>getchar (void)</b>	Legge il prossimo carattere dal flusso di input standard; equivalente a <b>getc(stdin)</b>
<b>gets (char *str)</b>	Legge i caratteri da <b>stdin</b> fino a <b>newline</b> o fino alla fine del file
<b>printf()</b>	Scrive uno o più valori in accordo con le regole di formazione definite dall'utente
<b>putchar (int ch)</b>	Scrive un carattere sul flusso di output standard; equivalente a <b>putc(stdout)</b>

## I/O su *stdin* e *stdout*

<b>puts (const char *str)</b>	Scrive una stringa di caratteri su <b>stdout</b> , inserendo un carattere di <b>newline</b> al termine della stringa
<b>scanf ()</b>	Legge uno o più valori da <b>stdin</b> , interpretandoli in accordo con le regole di formattazione definite dall'utente

## Fine file

<b>feof (FILE *stream)</b>	Controlla se è stata raggiunta la fine del file, associato allo <b>stream</b> , durante la precedente operazione di lettura
----------------------------	---

### Esempio

```
while (!feof(fp) )  
    getc(fp);
```

## I/O su file

<b><i>fclose( )</i></b>	Chiude un flusso
<b>fopen</b> (const char <i>*fname</i> , const char <i>*mode</i> )	Aprire un file, eventualmente creandolo, e vi associa un flusso. Richiede due argomenti: una stringa di caratteri che identifica il file e una descrizione della modalit� di apertura che determina le operazioni che possono essere compiute sul file.
<b>getc</b> (FILE <i>*stream</i> )	Legge un carattere da un flusso (macro)

## I/O su file

<b>fgetc</b> (FILE * <i>stream</i> )	Analoga a <i>getc( )</i> (funzione)
<b>fgets</b> (char * <i>str</i> , int <i>num</i> , FILE * <i>stream</i> )	Legge una stringa da un flusso di input specificato. Consente di specificare il numero massimo di caratteri da leggere



## I/O su file (2)

<b>putc (int ch, FILE *stream)</b>	Scrive un carattere su un flusso (macro)
<b>fputc (int ch, FILE *stream)</b>	Analoga a <b>putc()</b> (funzione)
<b>fputs (const char *str, FILE *stream)</b>	Scrive una stringa su un flusso di output specificato

## I/O su file (3)

<b>fprintf( )</b>	Scrive su un file specificato in modo analogo a quanto fa <i>printf</i> su <i>stdout</i>
<b>fscanf( )</b>	Legge i dati da un file specificato in modo analogo a quanto fa <i>fscanf</i> da <i>stdin</i>
<b>rewind (FILE *stream)</b>	Sposta l'indicatore di posizione del file <i>stream</i> all'inizio del file. Il file specificato da <i>stream</i> deve essere aperto.

```

/* Programma che legge caratteri dalla
   tastiera e li scrive su un file fino a che
   l'utente non immette il segno di '$'. */
#include <stdio.h>
#include <stdlib.h>
int main (void) {
    FILE *fp; char ch;
    if ((fp=fopen("prova.dat", "w")) !=
        NULL) {
        printf("Introduzione
caratteri\n");
        do {
            ch =getchar ( );
            putc(ch, fp);
        } while (ch != '$');
        fclose(fp); }
    else
        printf("Impossibile aprire il
file\n");
    return 0; }

```

## Esempio: conteggio caratteri in un file

```
#include <stdio.h>
int main( ) {
    FILE *fPtr ;
    char ch ;
    long int cont = 0 ;
    if (( fPtr = fopen( "provar" , "r" )) != NULL )
    {
        while( fscanf( fPtr , "%c" , &ch ) != EOF )
        {
            cont++ ; }
        printf( "cont = %ld\n" , cont ) ;
        /* cont = numero caratteri in "provar" */
        fclose( fPtr ) ;
    }
    else printf( "errore apertura file" ) ;
    return 0;
}
```

## Esempio: copia di un file

```
...  
FILE *frPtr , *fwPtr;  
char nfr[21] , nfw[21] , ch ;  
...  
if ((( frPtr = fopen( nfr , "r" )) != NULL ) &&  
    (( fwPtr = fopen( nfw , "w" )) != NULL )) {  
    while( fscanf( frPtr , "%c" , &ch ) != EOF )  
        fprintf( fwPtr , "%c" , ch ) ;  
    fclose( frPtr ) ;  
    fclose( fwPtr ) ; }  
else printf( "errore apertura file" ) ;  
...
```

## Esempio: acquisizione array da file (1)

```
...
FILE *frPtr ;
int v[ max_d ] , pos ;
...
/* Formato "comp.dat": 1 int per linea; numero
linee = max_d */

if ((( frPtr = fopen( "comp.dat" , "r" )) != NULL ))
{
    for( pos = 0 ; pos < max_d ; pos++ )
        fscanf( frPtr , "%d" , &v[ pos ] ) ;
    ...
    fclose( frPtr ) ;
}
else printf( "errore apertura file" ) ;
...
```

## Esempio: acquisizione array da file (2)

```
...
FILE *frPtr ;
int v[ max_d ] , pos = 0 ;
...
/* Formato "comp.dat": 1 int per linea; max
numero      linee = max_d */

if ((( frPtr = fopen( "comp.dat" , "r" )) != NULL ))
{
    while( fscanf( frPtr , "%d" , &v[ pos ] ) !=
EOF )
        pos++ ;
    fclose( frPtr ) ;
}
else printf( "errore apertura file" ) ;
...
```

## Esempio: acquisizione array bidimensionale

```
/* Formato "comp2.dat": d2 int per linea (separati da spazio/tab); d1 linee */
```

```
...
```

```
FILE *frPtr ;
```

```
int v[ d1 ][ d2 ] , r , c ;
```

```
...
```

```
if ((( frPtr = fopen( "comp2.dat" , "r" )) != NULL )) {
```

```
    for( r = 0 ; r < d1 ; r++ )
```

```
        for( c = 0 ; c < d2 ; c++ )
```

```
            fscanf( frPtr , "%d" , &v[ r ][ c ] ) ;
```

```
        ...
```

```
        fclose( frPtr ) ;
```

```
    }
```

```
else printf( "errore apertura file" ) ;
```

```
...
```