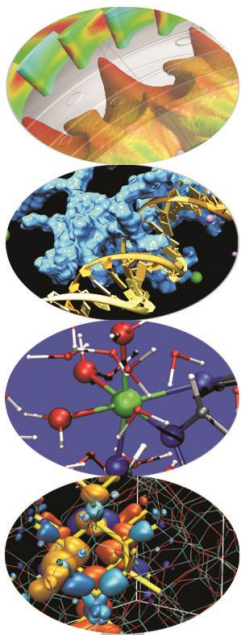
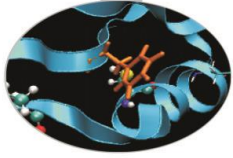


File in C

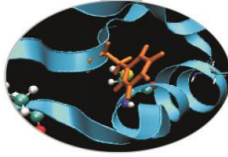


Indice



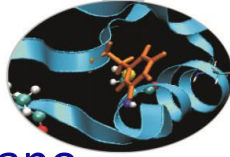
- **La gestione dei file in C e gli stream**
- **Apertura e chiusura di un file**
- **Operazioni sui file**
- **Accesso sequenziale e non sequenziale**

Gestione dei file in C



- In C all'interno della standard library vi è un header predisposto alla gestione dei file: `<stdio.h>`
- In C come in C++ vengono usate delle astrazioni per effettuare le operazioni di I/O, detti streams.
- Uno stream della `stdio.h` è rappresentato da un puntatore a FILE.
- Le funzioni presenti in questo header manipolano i dati contenuti nel FILE tramite questo puntatore.

Streams

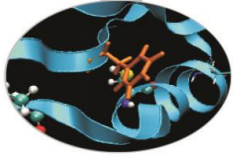


Ogni stream possiede delle proprietà che definiscono quali funzioni possano essere usate e come; molte di queste proprietà sono definite nel parametro di modalità di apertura del file:

- accesso: specifica se il file può essere letto scritto oppure letto/scritto
- testo/binario: i file di testo hanno le linee delimitate da caratteri speciali (EOL) ed il file stesso è chiuso da un altro carattere speciale (EOF). Un file binario è un file in cui ogni byte è gestito come un singolo carattere.
- buffer: stream bufferizzati ottimizzano la rapidità di lettura scrittura.

Ogni stream ha degli indicatori che ne specificano lo stato:

- indicatori di errore: viene settato quando un errore occorre in una operazione legata allo stream
- EOF: end of file
- indicatore di posizione: puntatore interno al file che indica la posizione in cui verrà letto o scritto il carattere successivo.

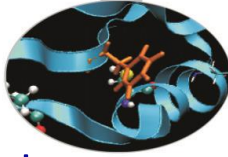


Streams predefiniti

Ogni volta che si include l'*header* per la gestione dei file vengono aperti 3 stream predefiniti:

- *stdin*: standard input, di default lo standard input corrisponde alla tastiera
- *stdout*: standard output, di default lo standard output è diretto sul video
- *stderr*: standard error, di default coincide con *stdout* ma può essere rediretto sul file di log

Apertura/Chiusura



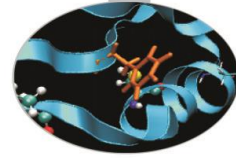
Utilizzando la struttura dati **FILE** definita in *stdio.h* e in particolare un puntatore ad essa, è possibile aprire/chiedere un file:

- apertura: `variabile_puntatore = fopen (const char * filename, const char * mode)`
- chiusura: `fclose(FILE *stream)`

```
#include <stdio.h>
FILE *punt;
punt=fopen("miofile","r");
fclose(punt);
```

- Le modalità con cui può essere aperto un file di testo sono:
 - "r": lettura, il file deve già esistere
 - "w": scrittura, il file viene creato se non esiste e sovrascritto in caso contrario
 - "a": scrittura in coda al file, se il file non esiste viene creato.
 - "r+": lettura + scrittura
 - "w+": scrittura+lettura
 - "a+": lettura+scrittura in coda

Apertura/Chiusura



Nel caso si debba lavorare con files binari le modalità di apertura sono le seguenti:

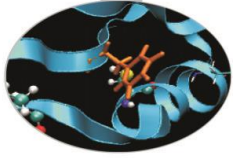
- "rb": lettura, il file deve già esistere
- "wb": scrittura, il file viene creato se non esiste e sovrascritto in caso contrario
- "ab": scrittura in coda al file, se il file non esiste viene creato.
- "r+b": lettura + scrittura
- "w+b": scrittura+lettura
- "a+b": lettura+scrittura in coda

E' possibile accedere in lettura o scrittura ai dati di un file operando su un intero blocco di dati testuali o binari di qualsiasi dimensione. Le funzioni utilizzate sono `fread()` e `fwrite()` i cui prototipi sono:

```
size_t fread ( void * ptr, size_t size, size_t count, FILE * stream );
```

```
size_t fwrite ( const void * ptr, size_t size, size_t count, FILE * stream );
```

Esempio

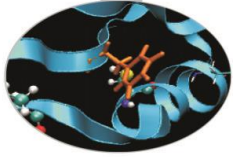


```
#include <stdio.h>

FILE *punt;

/* apertura del file */
punt=fopen("miofile","r"); //ritorna puntatore a NULL in caso di errore
if (punt==NULL)
{
    printf("errore in apertura del file /path/miofile \n");
    exit(1);
}

/*chiusura del del file*/
fclose(punt);
```

Operazioni sui files

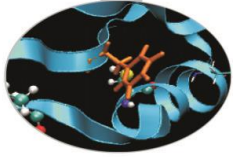
Una volta aperto un file può essere letto, scritto oppure entrambi.

Di seguito discuteremo le funzioni atte ad eseguire queste operazioni:

La scrittura può essere eseguita in numerosi modi

```
FILE *fp;  
fp= fopen("miofile","w");  
/*scrittura di un carattere nel file*/  
char c;  
putc(c,fp); //funzione o macro  
fputc(c,fp); //funzione  
/*scrittura di una stringa nel file*/  
char *str;  
fputs(str,fp);
```

Esempio



```
/*scrittura di n elementi di dimensione dim*/  
char *s; /* puntatore alla zona di  
          memoria*/  
  
unsigned int n, dim;  
fwrite(s, dim, n, fp); //ritorna quanti dati vengono scritti  
/* questa funzione è tipicamente usata per eseguire scrittura NON  
   FORMATTATA (BINARIA)*/  
  
/*scrittura formattata; esattamente come printf*/  
  
fprintf(fp, formato, arg1, arg2, ..argn);  
int indx;  
float value;  
  
fprintf(fp, "%d,%f\n", indx, value);
```



Operazioni sui files

- La lettura può essere eseguita in numerosi modi:

```
FILE fp;  
fp=open("myfile","r");  
/*lettura di un carattere da file*/  
char c;  
c = getc(fp);           //funzione o macro  
c = fgetc(fp);         //funzione  
  
/*rilettura di un carattere da file*/  
ungetc(c,fp);  
  
/*lettura di una stringa da file*/  
char str[1000];  
int n;                  //numero di caratteri da leggere -1  
fgets(str,n, fp);
```



Operazioni sui files

/*lettura di n dati di dimensione dim*/

```
char *s;
```

```
unsigned int n, dim;
```

```
fread(s,dim,n,fp);
```

/* usato tipicamente per la lettura binaria NON FORMATTATA*/

/*lettura formattata; esattamente come scanf*/

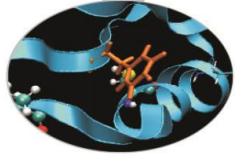
```
fscanf(fp,formato, &arg1,&arg2,..&argn);
```

```
int indx;
```

```
float value;
```

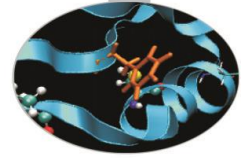
```
fscanf(fp, "%d,%f\n",&indx,&value);
```

Esempio



```
/*C standard I/O*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *input;
    char c, linea;
    int quanti;
    float costo;
    if ((input=fopen("dat.dat","r"))==NULL)
    {
        fprintf(stderr,"Impossibile aprire il file\n");
        exit(1);
    }
}
```

Esempio



```
while((c=getc(input))!=EOF)
{
    if(c!='*')
    {
        ungetc(c,input);
        fscanf(input,"%d:%f%c",&quanti,&costo,&linea);
        printf("Quantita' = %d\n" \
            "Costo (euro): %.3f\n" \
            "Totale (euro): %.3f\n" \
            "Linea produttiva: %c\n\n",quanti,costo,quanti*costo,linea);
    }
}
printf("EOF founded...exiting\n\n");
fclose(input);
return 0;
}
```



Esempio

OUTPUT:

Quantita'= 1

Costo (euro): 12.100

Totale (euro): 12.100

Linea produttiva: a

Quantita'= 2

Costo (euro): 13.100

Totale (euro): 26.200

Linea produttiva: b

Quantita'= 3

Costo (euro): 22.300

Totale (euro): 66.900

Linea produttiva: b

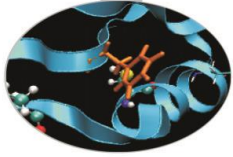
EOF founded...exiting

****dat.dat file****

1:12.1a

2:13.1b

3:22.3b



Accesso non sequenziale

Per muoversi in maniera non sequenziale all'interno di un file (cioè forzando dall'esterno il valore di offset all'interno del file) esistono le seguenti funzioni:

- *long int ftell (FILE * stream);*
- *int fseek (FILE * stream, long int offset, int origin)*
- *void rewind (FILE * stream)*

```
/*utilizzo e sintassi*/
```

```
FILE *fp;
```

```
long offset;
```

```
fseek(fp, 2, SEEK_CUR); /*sposta il valore di offset 2 bytes oltre  
la posizione corrente; altre opzioni per il terzo parametro  
sono l'inizio e la fine del file: SEEK_SET e SEEK_END */
```

```
rewind(fp); //riporta il valore di offset a zero
```

```
offset=ftell(fp); //ritorna il valore di offset  
dall'inizio del file
```