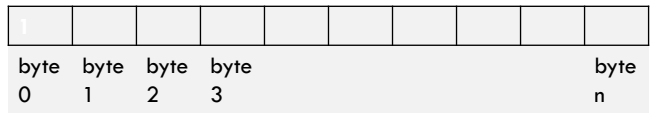
Gestione dei file in C

Generalità

- Il linguaggio C non contiene alcuna istruzione di Input/Output.
- Tali operazioni vengono svolte mediante chiamate a funzioni definite nella libreria standard contenute nel file *stdio.h*.
- Tali funzioni rendono possibile la lettura/scrittura in modo indipendente dalle caratteristiche proprie dei dispositivi di Input/Output.
- Le stesse funzioni possono essere utilizzate, ad esempio, sia per leggere un valore dalla tastiera sia per leggere un valore da un dispositivo di memoria di massa. Lo stesso vale per le funzioni di scrittura: le stesse operazioni possono essere utilizzate sia per la visualizzazione sullo schermo sia per scrivere un valore su un disco o una stampante.
- Ciò è possibile poichè il sistema di I/O C è caratterizzato da un'interfaccia indipendente dal dispositivo effettivo che si interpone tra il programmatore e il dispositivo. Tale interfaccia è chiamata flusso, mentre il dispositivo effettivo è chiamato file.

Flusso

- Il sistema di I/O C associa ad ogni dispositivo fisico un dispositivo logico chiamato flusso.
- Poichè tutti i flussi si comportano alla stessa maniera, possono essere utilizzate le stesse funzioni per la loro gestione.



File

- Un File è un qualsiasi dispositivo, da un disco a un terminale a una stampante.
- Per associare un flusso a un file è necessario un'operazione di apertura. Una volta aperto un file sarò possibile scambiare informazioni tra il file e il programma.
- Per eliminare l'associazione tra flusso e file è necessaria un'operazione di chiusura. Nel caso un file aperto in scrittura, l'eventuale contenuto del flusso viene scritto sul dispositivo fisico.

FILE

- Ogni flusso ha associato una struttura chiamata FILE contenente i seguenti campi:
  - modalità di utilizzo del file (lettura, scrittura o lettura e scrittura);
  - posizione corrente su file (indicante il prossimo byte da leggere o scrivere su file);
  - un indicatore di errore di lettura/scrittura;
  - un indicatore di end-of-file, indicante il raggiungimento della fine del file.

- Ogni operazione di apertura a file restituisce un puntatore a una variabile di tipo FILE. Per potere leggere e scrivere i file è necessario usare delle variabili di tipo puntatore a file, dichiarate nel seguente modo:

```
FILE *pf;
```

fopen

- L'apertura di un file viene realizzata mediante la funzione fopen avente il seguente prototipo:
  `FILE *fopen(const char *nomefile, const char *modalità);`

- dove *nomefile* è una stringa di caratteri indicante il nome del file da aprire e *modalità* indica il modo in cui il file deve essere aperto.
- Se si verifica un errore in apertura del file, la fopen() restituisce un puntatore nullo.

fopen

- Volendo, ad esempio, aprire in scrittura il file *test.txt* scriveremo:

```
FILE *pf;
pf=fopen("test.txt",''w'');
if(pf==NULL)
    printf(``Impossibile aprire il file'');
else
```

- In generale, prima di accedere ad un file occorre assicurarsi che la chiamata a fopen() sia stata eseguita con successo.

Modalità di apertura

MODALITA'	SIGNIFICATO
r	Apre un file di testo in lettura
w	Crea un file di testo in scrittura
a	Apre un file di testo in modalità append
r+	Apre un file di testo in lettura/scrittura
w+	Crea un file di testo in lettura/scrittura
a+	Crea o apre un file di testo in modalità append per lettura/scritura

fclose

•La chiusura di un file viene realizzata mediante la funzione fclose avente il seguente prototipo:

```
int fclose(FILE *pf);
```

dove *pf* è il puntatore restituito dalla fopen().

fread, fwrite

•Le funzioni fread() e fwrite() vengono utilizzate per la lettura e la scrittura su file di un blocco di dati di qualsiasi dimensione.

•Il loro prototipo è il seguente:

```
size_t fread (void *buffer, size_t n_byte, size_t num,FILE *pf);
size_t fwrite(const void *buffer, size_t n_byte, size_t num, FILE *pf);
```

Per fread() *buffer* è un puntatore alla regione di memoria che riceverà i dati letti dal file.

Per fwrite() *buffer* è un puntatore alla regione di memoria che contiene i dati da scrivere sul file.

Il valore di *num* determina il numero di oggetti di dimensione *n_byte* da leggere/scrivere

Il tipo size_t è definito in STDIO.H e si tratta di un intero unsigned

Esempio1 fread, fwrite (1/2)

```
#include <stdio.h>

main()
{ FILE *pfile;
  long matricola,m;
  int nesami,n;

  pfile=fopen("dati.dat","w");
  if(pfile!=NULL)
     { printf("Matricola: ");
       scanf("%ld",&matricola);
       printf("N. esami : ");
       scanf("%d",&nesami);
       fwrite(&matricola,sizeof(long),1,pf);
        fwrite(&nesami,sizeof(int),1,pf);
       fclose(pfile);
     }
  else printf("Errore in scrittura\n");
```

Esempio1 fread, fwrite (2/2)

```
pfile=fopen("dati.dat","r");
  if(pfile) {
              fread(&m,sizeof(long),1,pf);
              fread(&n,sizeof(int),1,pf);
              printf("Matricola: %ld\n",m);
              printf("N.Esami  : %d\n",n);
              fclose(pfile);
            }
 else printf("Errore in lettura");
}
```

Esempio2 con fwrite() e fread() (1)

```
#include <stdio.h>

struct studente {
   long matricola;
   int esami;   };

void stampa_file( FILE *pf);

int main()
{ FILE *pfile;
  struct studente stud;

  pfile=fopen("dati.dat","w");
  if(pfile)
     {  printf("Matricola: ");
        scanf("%ld",&stud.matricola);
       printf("N. esami : ");
       scanf("%d",&stud.esami);
       fwrite(&stud,sizeof(stud),1,pfile);
        fclose(pfile);
   }
  else printf("Errore in scrittura\n");
```

Esempio2 con fwrite() e fread() (2)

```
  pfile=fopen("dati.dat","r");
  if(pfile!=NULL)
        {
            stampa_file(pfile);
            fclose(pfile);
         }
 else printf("Errore in lettura");
}


void stampa_file( FILE *pf)
{
  struct studente a;

  fread(&a,sizeof(struct studente),1,pf);
  printf("Matricola: %ld\n",a.matricola);
  printf("N.Esami  : %d\n",a.esami);
}
```

Esempio3 con fwrite() e fread() (1)

```
#include <stdio.h>

void leggi_vettore(int *VI,int n);
void stampa_vettore (int *VF, int m);

int main()
{ FILE *pfile;
  int V[10],VF[10],i,n;

  printf("Quanti numeri ? (<=10)\n");
  scanf("%d",&n);
  leggi_vettore(V,n);
  pfile=fopen("dati.dat","w");
  if(pfile)
     { fwrite(V,sizeof(int),n,pfile);
       fclose(pfile);
   }
  else printf("Errore in scrittura\n");
  pfile=fopen("dati.dat","r");
  if(pfile)
    { fread(VF,sizeof(int),n,pfile);
      stampa_vettore(VF,n);
       fclose(pfile);
   }
 else printf("Errore in lettura");
}
```

Esempio3 con fwrite() e fread() (2)

```
void leggi_vettore(int *VI,int n)
{  int i;

   for(i=0;i<n;i++)
    { printf("Nuovo numero: ");
          scanf("%d",&VI[i]);
       }
}

void stampa_vettore (int *VF, int m)
{  int i;

   for(i=0;i<m;i++)
      printf("V[%d]: %d\n",i,VF[i]);
}
```

Esempio4 con fwrite() e fread() (1)

```
#include <stdio.h>

struct studente {
   long matricola;
   int esami;
};

void leggi_vettore(struct studente *VA, int *pn);
void salva_file (struct studente *VA, int n);
void stampa_file ();

int main()
{
 struct studente VA[10];
 int numero_letti;

leggi_vettore(VA,&numero_letti);
 salva_file(VA,numero_letti);
 stampa_file();

}
```

Esempio4 con fwrite() e fread() (2)

```
void leggi_vettore(struct studente *VA, int *pn)
{  int i;
   printf("Quantielementi ?\n");
   scanf("%d",pn);
   for(i=0;i<*pn;i++)
    { printf("Matricola: ");
          scanf("%ld",&VA[i].matricola);
      printf("Numero esami: ");
      scanf("%d",&VA[i].esami);
       }

}
void salva_file (struct studente *VA, int n)
{  FILE *pf;
   int i;
   pf=fopen("d:\\temp\\stud.txt","wb");
   if(pf)
    {
     for(i=0;i<n;i++)
    fwrite(&VA[i],sizeof(struct studente),1,pf);
     fclose(pf);
    }
   else printf("Errore in scrittura");
}
```

Esempio4 con fwrite() e fread() (3)

```
void stampa_file ()
{
   FILE *pf;
   int i,n;
   struct studente ax;

   pf=fopen("d:\\temp\\stud.txt","rb");
   if(pf)
    {
     do
      {n=fread(&ax,sizeof(struct studente),1,pf);
       if(n!=0)
        {printf("Matricola : %ld\n",ax.matricola);
         printf("Num. esami: %d\n",ax.esami);
        }
      } while (n!=0);
     fclose(pf);
    }
   else printf("Errore in lettura");
}
```

Esempio5 con fwrite() e fread() (1)

```
#include <stdio.h>

void inizializza(int *V, int *pn);
void visualizza_vettore(int *V, int n);
void aggiungi_elementi(int *V, int *pn);
void salva_vettore(int *V, int n);

int main()
{ int V[30],i,n;

  inizializza(V,&n);
  visualizza_vettore(V,n);
  aggiungi_elementi(V,&n);
  salva_vettore(V,n);
}
```

Esempio5 con fwrite() e fread() (2)

```
void inizializza(int *V, int *pn)
{ FILE *pf;
  int i=0;

  pf=fopen("dati.dat","r");
  if(pf)
  {  while (fread(&V[i],sizeof(int),1,pf)>0 && i<30)
       i++;
    fclose(pf);
  }
 else printf("Nessun elemento");
 *pn= i;
}
```

Esempio5 con fwrite() e fread() (3)

```
void visualizza_vettore(int *V, int n)
{ int i;
  for(i=0;i<n;i++)
   printf("%d\n",V[i]);
}
void aggiungi_elementi(int *V, int *pn)
{ int al,i;
  printf("Quanti elementi aggiungere ? (<= %d)",30-*pn);
  scanf("%d",&al);
  for(i=*pn;i<*pn+al;i++)
   {printf("Numero : ");
    scanf("%d",&V[i]);
   };
  *pn=*pn+al;
}
void salva_vettore(int *V, int n)
{ FILE *pf;
  pf=fopen("dati.dat","w");
  if(pf)
    fwrite(V,sizeof(int),n,pf);
  else printf("errore in scrittura\n");
}
```

fseek()

Questa funzione permette l'accesso diretto al file in lettura o in scrittura impostando la posizione dell'indicatore del file.

Il suo prototipo è il seguente:

```
int fseek(FILE *pf, long n_byte, int origine);
```

dove

n_byte indica la posizione, espressa come numero di byte, rispetto all'origine in cui si vuole portare l'indicatore di posizione del file

Origine può assumere uno dei seguenti valori:

SEEK_SET inizio del file (0)

SEEK_CUR posizione corrente (1)

SEEK_END fine del file (2)

ftell()

Questa funzione restituisce la posizione corrente rispetto all'inizio del file, espressa come numero di byte.

Il suo prototipo è il seguente:

```
long ftell(FILE *pf);
```

Esempio con fseek() e ftell() (1)

```
#include <stdio.h>

void leggi_salva(FILE *pf);
long dim_file (FILE *pf);
void stampa_vettore (int *VF, int m);

int main()
{ FILE *pfile;
  int V[30];
  long numero;

  pfile=fopen("dati.dat","w");
  if(pfile)
    { leggi_salva(pfile);
      fclose(pfile);
    }
  else printf("errore");
  pfile=fopen("dati.dat","r");
  if(pfile)
    { numero=dim_file(pfile)/sizeof(int);
      fread(V,sizeof(int),numero,pfile);
      stampa_vettore(V,numero);
      fclose(pfile);
   }
 else printf("Errore in lettura");
}
```

Esempio con fseek() e ftell() (1)

```
void leggi_salva(FILE *pf)
{ int a;
   do
   { printf("Inserisci un numero");
     scanf("%d",&a);
     fwrite(&a,sizeof(a),1,pfile);
   } while (a != 0);
}

long dim_file (FILE *pf)
{
  long pos_corrente,lunghezza;

  pos_corrente=ftell(pf);
  fseek(pf,0,SEEK_END);
  lunghezza=ftell(pf);
  fseek(pf,pos_corrente,SEEK_SET);
  return lunghezza;
}


void stampa_vettore (int *VF, int m)
{
   int i;

   for(i=0;i<m;i++)
     printf("V[%d]: %d\n",i,VF[i]);
}
```

feof()

- La funzione feof() restituisce un valore logico vero nel caso in cui è raggiunta la fine del file e zero in tutti gli altri casi.
- Il prototipo della feof è il seguente:

```
int feof(FILE *pf)
```

fscanf, fprintf

- Le funzioni fscanf() e fprintf() vengono utilizzate per la lettura e la scrittura su file. Il loro comportamento è lo stesso delle funzioni scanf() e printf().
- Il loro prototipo è il seguente:

```
int fscanf(FILE *pf, const char *stringa_controllo, ..);

int fprintf(FILE * pf, const char *stringa_controllo,..);
```

Esempio: Lettura da file e visualizzazione sullo schermo

```
#include <stdio.h>
int main()
{ FILE *pf;
  int a;
  pf=fopen("numeri.txt","r");
  if(pf)
 {
   while(!feof(pf))
   { fscanf(pf,"%d\t",&a);
     printf("%d\n",a);
   }
   fclose(pf);
  }
  else
  printf("errore");
}
```

Esempio: Lettura da tastiera e scrittura su file

```
#include <stdio.h>
int main()
{ int num,a;
  FILE *pf;
  printf("Quanti numeri vuoi inserire ?");
  scanf("%d",&num);
  pf=fopen("dati.txt","w");
  if(pf)
  { for(;num>0;num--)
    { printf("Inserisci un nuovo numero: ");
      scanf("%d",&a);
      fprintf(pf,"%d\t",a);
    }
    fclose(pf);
  }
 else printf("Errore");
}
```

Scrittura e lettura delle informazioni (cognome, nome, matricola, media) relative a n studenti (1)

```
#include <stdio.h>

int main()
{ FILE *pf;
  char cognome[20],nome[20];
  long matricola;
  double media;
  int i,nstud;

  printf("Quanti studenti ?\n");
  scanf("%d",&nstud);
  pf=fopen("studenti.txt","w");   /* Apertura file*/
  if(pf==NULL) printf("Errore in scrittura\n");
  else  /* file aperto con successo */
   {  for(i=0;i<nstud;i++)
      { printf("Cognome:"); scanf("%s",cognome);
        fprintf(pf,"Cognome: %s\n",cognome);
        printf("Nome: "); scanf("%s",nome);
        fprintf(pf,"Nome: %s\n",nome);
        printf("Matricola: "); scanf("%ld",&matricola);
        fprintf(pf,"Matricola: %ld\n",matricola);
        printf("Media: "); scanf("%lf",&media);
        fprintf(pf,"Media: %lf\n",media);
      }
      fclose(pf);
    }
```

Scrittura e lettura delle informazioni (cognome, nome, matricola, media) relative a n studenti (2)

```
/* lettura da file */
   pf=fopen("studenti.txt","r");
  if(pf==NULL)
     printf("Errore in lettura\n");
  else
   { while (!feof(pf))
       {    fscanf(pf,"Cognome: %s\n",cognome);
            printf("Cognome: %s\n",cognome);
            fscanf(pf,"Nome: %s\n",nome);
            printf("Nome: %s\n",nome);
            fscanf(pf,"Matricola: %ld\n",&matricola);
            printf("Matricola: %ld\n",matricola);
            fscanf(pf,"Media: %lf\n",&media);
            printf("Media: %lf\n",media);
       }
     fclose(pf);
    }
}
```