Gestione dei File

Prof. Salvatore Venticinque
Prof. Mario Magliulo

Il File

- Unità di memorizzazione dati da parte del FileSystem
- Contiene informazioni conservate in maniera non volatile nelle memorie di massa
- Dati sono salvati come byte

- **Tipologia**:
  - *Testo*:
    - Ogni byte è la rappresentazione ASCII di un carattere
  - *Binari*:
    - Ogni byte è la rappresentazione dell'informazione specifica dell'applicazione (Word, Excel, ….)

Operazioni su File

- **Apertura**:
  - Viene creato un canale tra il programma e l'istanza del file nel filesystem
- **Scrittura**:
  - Vengono inviati bytes dal programma al filesystem attraverso il canale
- **Lettura**:
  - Vengono inviati bytes dal filesystem al programma
- **Chiusura**:
  - Viene svuotato il canale e rilasciate le risorse utilizzate per la gestione del file

In particolare si tenga conto che:

- il sistema operativo si occupa di soddisfare le richieste del programma
- per ovviare alla lentezza delle memorie di massa, il sistema operativo effettivamente scrive nei momenti in cui l'elaboratore è più scarico, eventualmente evadendo più richieste allo stesso tempo.

File nel linguaggio C

Nel linguaggio C:

- I file costituiti da sequenze lineari di byte
- Sono disponibili soltanto alcune funzioni di base per operare sui file

- E possibile:
  - Creazione,
  - rimozione di file,
  - lettura e scrittura di byte da/su file

Le funzioni di libreria

Le funzioni e le strutture dati per la gestione dei file

- Fornite da ***stdio.h***
  - fopen
  - fread
  - fwrite
  - fclose
  - fprintf
  - fscanf
  - fputs
  - fgets
  - fputc
  - fgetc

Apertura del file

```
#include <stdio.h>
main()
{
  FILE *fp;
  fp = fopen("ordini", "r");
   ...
```

Apertura del file

- La funzione fopen apre il file di nome ordini;
- L'operazione di apertura restituisce la variabile puntatore fp (*file pointer*), al file di nome ordini.
- Il file pointer fp per viene utilizzato per le successive operazioni
- Nel caso in cui si verifichi un errore, per esempio perché il file ordini non esiste, la funzione fopen ritorna un file pointer ***NULL*.**

Apertura del file

In generale, le possibili modalità di apertura di un file, specificate dal secondo parametro di fopen, sono le seguenti.

- **"r": sola lettura** – sarà possibile eseguire soltanto operazioni di lettura. Se il file non esiste la funzione fopen ritornerà il codice di errore NULL.
- **"w": sola scrittura** – sarà possibile eseguire solamente operazioni di scrittura. Se il file non esiste sarà automaticamente creato, in caso contrario il contenuto del file preesistente sarà perso.
- **"r+": lettura e scrittura** – sarà possibile eseguire operazioni sia di lettura sia di scrittura. Se il file non esiste la funzione fopen ritornerà il codice di errore NULL.
- **"w+": scrittura e lettura** – sarà possibile eseguire operazioni sia di scrittura sia di lettura .
- **"a": append** – sarà possibile eseguire soltanto operazioni di scrittura. Tutte le scritture verranno sempre eseguite in coda al contenuto attuale. Se il file non esiste verrà creato.
- **"a+": lettura e append** – sarà possibile eseguire operazioni sia di lettura sia di scrittura. Se il file non esiste verrà automaticamente creato, in caso contrario il contenuto del file preesistente verrà mantenuto.

Chiusura File

Se, e solo se, il file è stato aperto correttamente, occorre chiudere il file.

```
int main()
{
FILE *fp;
fp = fopen("ordini", "r");
if (fp!=NULL)
   {
      [....]
      fclose(fp);
        }
```

Chiusura File

- La chiusura del file garantisce che tutti i dati scritti nel file ordini siano salvati su disco;

- Il sistema operativo, per ottimizzare le prestazioni del programma, ritarda le scritture sulla memoria di massa mantenendo le informazioni temporaneamente in memoria centrale.

Scrittura su file

n = fwrite(buf, dimensione, elementi, fp);

fwrite scrive byte del vettore buf nel file puntato da fp.

I parametri della funzione sono i seguenti:
buf: il vettore che contiene i dati che devono essere memorizzati nel file
dimensione: rappresenta la dimensione in byte di un elemento del vettore
elementi: indica il numero di elementi del vettore
fp: è il puntatore al file dove devono essere memorizzati i dati

n rappresenta il numero di elementi effetivamente scritti nel file.
n < elementi se il file ha raggiunto la dimensione massima
n = -1 se c'è stato qualche errore (es. Il file non è stato aperto)

Memorizzazione array di interi

```
/* Scrittura di una stringa in un file */
#include <stdio.h>
#include <string.h>
Int main()
{
  int buf[100];
  FILE *fp;
  int len;
  /* Buffer */
  /* File pointer */
  /* Legge da tastiera la dimensione del vettore*/
  printf("Inserisci un intero : ");
  scanf("%d",&len);
  for(int i=0;i<len;i++)
     scanf("%d",&buf[i]);
  fp = fopen("numeri", "w"); /* Crea il file di numeri */
  if(fp!=NULL)
    {/* Memorizza gli elementi del vettore nel file */
      fwrite(buf, 4, len, fp);
      fclose(fp);
      /* Chiude il file */
     }
}
```

```c
/* Scrittura di una stringa in un file */
#include <stdio.h>
#include <string.h>
Int main()
{
  char buf[100];
  FILE *fp;
  int len;
  /* Buffer */
  /* File pointer */
  /* Legge da tastiera il nome del fornitore */
  printf("Inserisci un fornitore : ");
  scanf("%s",buf);
  len = strlen(buf);
  fp = fopen("fornitori", "w"); /* Crea il file fornitori */
  if(fp!=NULL)
    {/* Memorizza il nome del fornitore nel file */
      fwrite(buf, 1, len, fp);
      fclose(fp);
      /* Chiude il file */
     }
}
```

Lettura da File

n = fread(buf, dimensione, elementi, fp);

- **buf**: è il vettore dove devono essere trasferite le informazioni lette dal file
- **dimensione**: rappresenta la dimensione in byte di un elemento del vettore
- **elementi**: indica il numero di elementi del vettore
- **fp**: punta al file da leggere

La funzione fread legge dati dal file fp e li trasferisce nel vettore buf.

Esempio

```
int elementi, dimensione;
char buf[100];
FILE *fp;
int n;
...
elementi = 100;
dimensione = 1;
n = fread(buf, dimensione, elementi, fp);
```

- Il valore di ritorno della fread indica il numero di elementi letti dal file;
- tale numero può non coincidere con il numero di elementi del vettore buf,
  - nel caso in cui il file sia vuoto e
  - nel caso in cui contenga un numero di elementi inferiore a quello di buf.
- Se il valore di ritorno assume valore negativo significa che è stato commesso qualche errore, per esempio il file non è stato aperto.
- Per file grandi (quasi tutti):
  - fread deve essere ripetuta più di una volta, leggendo a ogni chiamata soltanto una limitata porzione del file.

- La operazioni di lettura accedono al file in maniera sequenziale e mantengono traccia del punto in cui si è arrivati nella lettura.
- A ogni chiamata a fread il puntatore si sposta in avanti di un numero di byte pari a quelli che sono stati letti.
- Quando tutto il contenuto del file è stato letto la funzione fread ritorna il valore 0.
- Ogni ulteriore tentativo di lettura fallirà e fread continuerà a restituire il valore di ritorno 0.

Conteggio caratteri in file

```c
/* Determina il numero di caratteri di un file esistente */
#include <stdio.h>
int main()
{
	char buf[100];
	FILE *fp;
	long nc, n;
	int fine_file =0;

	fp = fopen("ordini", "r");/* Apertura del file clienti */
	if( fp == NULL )
	printf("Errore : il file ordini non esiste\n");
	else {
		nc = 0;
		do {
			n = fread(buf, 1, 100, fp);
			if(n==0)
			fine_file = 1;
			nc += n;
		}
	 while(fine_file==0);
	fclose(fp);
	printf("Il file clienti contiene %ld caratteri\n", nc);
	}
}
```

Copia di un file in un altro

```
#include <stdio.h>
Int main()
{
	FILE *fpin, *fpout;
	char buf[512]; int n;
	fpin = fopen("ordini","r");
	if(fpin!=NULL) {
		fpout = fopen("ordini.bak", "w");
		if(fpout!=NULL) {
			do{
				n = fread(buf, 1, 512, fpin);
				if( n != 0 )
					fwrite(buf, 1, n, fpout);
			}while(n>0);
			fclose(fpin); fclose(fpout);
		}
		else { printf("Il file ordini.bak non può essere aperto\n");
		fclose(fpin); }
	}
	else printf("Il file ordini non esiste\n");
}
```

Lettura random nel File

- In C è possibile operare sui file di byte anche con modalità random.
- La funzione fseek consente infatti di muovere il puntatore di lettura e/o scrittura in una qualunque posizione.

err = fseek(fp, n, mode);

I parametri della funzione fseek hanno il seguente significato:

- **fp**: è il file pointer;
- **n**: indica di quanti byte il file pointer deve essere spostato; se n è negativo significa che il file pointer deve essere spostato indietro invece che in avanti;
- **mode**: indica a partire da quale posizione muovere il file pointer; se mode vale 0 significa che lo spostamento è a partire dall'inizio, se vale 1 è dalla posizione corrente e se, infine, vale 2 è a partire dalla fine del file

- Il valore di ritorno della funzione fseek è negativo se si è verificato un errore, maggiore o uguale a 0 in caso contrario.

Esempi

Come spostare il puntatore:

- All'inizio?
- Alla fine?
- A 10 byte dall'inizio?
- A 10 byte dalla fine?
- A 10 byte dopo la posizione corrente?
- A 10 byte prima della posizione corrente?

ftell()

- Il C mette a disposizione una funzione per conoscere la posizione corrente del file pointer;

n = ftell(fp);

- La funzione ftell ritorna la posizione corrente del file pointer;
- se si verifica un errore, per esempio se il file non è stato aperto, ftell ritorna un valore negativo.

Esempio

```
#include <stdio.h>
int main(int argc, char **argv)
{
        FILE *fp;
        long n;
   if( argc < 2 )
        printf("File non specificato\n");
        else {
                fp = fopen(argv[1], "r");
                if( fp != NULL ) {
                   fseek(fp,0,2);
                   n = ftell(fp);
                   fclose(fp);
                   printf("La dimensione del file è %ld\n", n);
                 }
                else
                     printf("Errore : il file %s non esiste\n", argv[1]);
            }
}
```