Gestione dei File in C

Maurizio Palesi
DIIT—Università di Catania
Viale Andrea Doria 6, 95125 Catania
mpalesi@diit.unict.it
http://www.diit.unict.it/users/mpalesi

Sommario

In questo documento saranno introdotte le funzioni per la gestione dei file in C. L'importanza dei file risiede nel fatto che per mezzo di esso è possibile conservare grosse quantità di dati che debbano essere utilizzate in tempi diversi o che debbano essere aggiornate.

Indice

1 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.

2 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.

Esistono due tipi di flussi: flussi binari e di testo. Un flusso binario è formato da una sequenza di byte con una corrispondenza uno ad uno con i byte presenti sul dispositivo fisico. Un flusso di testo è una sequenza di

Modalita	*Descrizione*
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
rb	Apre un file binario in lettura
wb	Crea un file binario in scrittura
ab	Apre un file binario 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
r+b	Apre un file binario in lettura/scrittura
w+b	Crea un file binario in lettura/scrittura
a+b	Crea o apre un file binario in modalità append per lettura/scrittura

Tabella 1: Modalità di apertura di un file.

caratteri generalmente suddivisa in linee terminate da un carattere di *newline*. In un flusso di testo alcuni caratteri potrebbero essere tradotti. Ad esempio il 30.7199 0 Td(in) 3222 0 0 -4.8 144-4.8 144-42 0 Td(flusso)Tj 23399 0 Td(/BP(file.)Tj /può

```
FILE *pf;

pf = fopen("test", "w");
if (pf == NULL)
  {
    printf("Impossibile aprire il file");
    exit(1);
  }
```

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

3.2 `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()`.

3.3 `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);
```

3.4 `fscanf()` e `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_di_controllo .c.c;
                                                    int
```

```
      }
      fclose(pf);
    }
   else
     printf("errore durante la l'apertura del file.");
}
```

□

Esercizio 3.2 ()

Lettura da tastiera e scrittura su file.

Risoluzione

```
#include <stdio.h>

void main(void)
{
  int num, a;
  FILE *pf;

  printf("Quanti numeri vuoi inserire? ");
  scanf("%d", &num);

  pf = fopen("dati.txt", "w");
  if (pf)
  {
    for( ; num ; num--)
    {
      printf("Inserisci un nuovo numero: ");
      scanf("%d", &a);
      fprintf(pf, "%d\t", a);
    }
    fclose(pf);
  }
  else
    printf("Errore durante l'apertura del file.");
}
```

□

Esercizio 3.3 ()

Lettura da file e visualizzazione sullo Standard Output di stringhe.

Risoluzione

```
#include <stdio.h>

void main(void)
{
  FILE *pf;
```

```
char nome[30], cognome[20];
int  esami;

pf = fopen("dati.txt", "r");
if (pf)
  {
    while(!feof(pf))
      {
        fscanf(pf, "%s\t", nome);
        fscanf(pf, "%s\t", cognome);
        fscanf(pf, "%d\t", &esami);
        printf("Nome: %s\tCognome: %s\tNum.esami: %d\n",
               nome, cognome, esami);
      }
    fclose(pf);
  }
else
  printf("Errore duranteo
```

```
}
```

□

Esercizio 3.5 ()

Massimo tra numeri letti da file (Versione con funzioni).

Risoluzione

```
#include <stdio.h>

int MaxDaFile(char *nomefile, int *max);

void main(void)
{
  int massimo;

  if (MaxDaFile("numeri.txt", &massimo))
    printf("Il massimoè  :%d", massimo);
  else
    printf("Errore su file");
}

int MaxDaFile(char *nomefile, int *max)
{
  FILE *pf;
  int  a;

  pf = fopen(nomefile, "r");
  if (pf)
    {
      if (!feof(pf))
        fscanf(pf, "%d\t", max);
      while (!feof(pf))
        {
          fscanf(pf, "%d\t", &a);
          if (*max < a)
            *max = a;
        }

      fclose(pf);
      return 1;
    }
  else
    return 0;
}
```

□

Esercizio 3.6 ()

Lettura da tastiera di un vettore di struct e scrittura su file.

Risoluzione

```c
#include <stdio.h>

struct Dato {
  long matricola;
  int  nmaterie;
};

void Leggi(struct Dato *d);
void ScriviSuFile(char *nomefile, struct Dato *vect);

void main(void)
{
  struct Dato vettore[10];
  int i;
```

4 Accesso a Blocchi

4.1 `fread()` e `fwrite()`

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

```
int fread(void *punt, int dimensione_elemento, int numero_elementi, FILE *pf);
int fwrite(void *punt, int dimensione_elemento, int numero_elementi, FILE *pf);
```

La funzione `fread()` legge un blocco di `numero_elementi`, ciascuno di `dimensione_elemento` byte, dal file cui fa riferimento il puntatore `pf` e li copia in memoria a partire dall'indirizzo indicato dal puntatore `punt`. La funzione restituisce il numero di elementi effettivamente letti; tale numero è inferiore al numero di elementi richiesti o perchè c'è stato un errore in lettura o perchè si è arrivati alla fine del file.

La funzione `fwrite()` scrive un blocco di `numero_elementi`, ciascuno di `dimensione_elemento` byte, sul file cui fa riferimento il puntatore `pf` prelevandoli dalla memoria a partire dall'indirizzo indicato dal puntatore `punt`. La funzione restituisce il numero di elementi effettivamente scritti; tale numero è inferiore al numero di elementi richiesti perchè c'è stato un errore in scrittura.

Esercizio 4.1 ()

Scrittura e successiva lettura di numeri interi su file.

Risoluzione

```
#include <stdio.h>

main()
{
  FILE *pf;
  int  i, num;

  pf = fopen("numeri.dat", "w");
  if (pf)
    {
      for(i=0; i<4; i++)
        {
          printf("Inserisci un nuovo numero: ");
          scanf("%d", &num);
          fwrite(&num, sizeof(in"'d(,)Tj /R9 8.0 1 514.556 637.06 Tm(p)Tj 5.698 0 Td(num)Tjj 5.75996 0Td())T8
```

```
      fclose(pf);
    }
  else
    exit(1);
}
```

□

Esercizio 4.2 ()

Scrittura e successiva lettura di strutture su file (versione 1).

Risoluzione

```
#include <stdio.h>

struct Elemento {
  long matricola;
  int materie;
};

main()
{
  FILE *pf;
  int  i, num;
  struct Elemento el;

  pf = fopen("numeri.dat", "w");
  if (pf)
    {
      for (i=0; i<4; i++)
        {
          printf("Inserisci la matricola: ");
          scanf("%ld", &(el.matricola));
          printf("Inserisci il numero di esami superati: ");
          scanf("%d", &el.materie);
          fwrite(&el, sizeof(struct Elemento), 1, pf);
        }
      fclose(pf);
    }
  else
    exit(1);

  pf = fopen("numeri.dat", "r");
  if (pf)
    {
      for (i=0; i<4; i++)
        {
          fread(&el, sizeof(struct Elemento), 1, pf);
          printf("Matricola: %ld\tMaterie: %d\n", el.matricola, el.materie);
        }
      fclose(pf);
```

```
    }
  else
    exit(1);
}
```

□

Esercizio 4.3 ()

Scrittura e successiva lettura di strutture su file (versione 2).

Risoluzione

```
#include <stdio.h>

struct Elemento {
  long matricola;
  int materie;
};

main()
{
  FILE *pf;
  int  i, num;
  struct Elemento v[4], el;

  for (i=0; i<4; i++)
    {
      pf = fopen("numeri.dat", "a+");
      if (pf)
        {
          printf("Inserisci la matricola: ");
          scanf("%ld", &el.matricola);
```

```
    printf("Matricola: %ld\tMaterie: %d\n", v[i].matricola, v[i].materie);
}
```

□

Esercizio 4.4 ()

Scrittura e successiva lettura di strutture su file (versione 3).

Risoluzione

```
#include <stdio.h>

struct Elemento {
  long matricola;
  int  materie;
};

main()
{
  FILE *pf;

  int             i, num, okay;
  struct Elemento el;

  printf("Quanti elementi vuoi inserire? ");
  scanf("%d", &num);

  for(i=0; i<num; i++)
    {
      pf = fopen("numeri.dat", "a+");
      if (pf)
        {
          printf("Inserisci la matricola: ");
          scanf("%ld", &el.matricola);
          printf("Inserisci il numero di esami superati: ");
          scanf("%d", &el.materie);
          fwrite(&el, sizeof(struct Elemento), 1, pf);
          fclose(pf);
        }
      else
        exit(1);
    }

  pf = fopen("numeri.dat", "r");
  if (pf)
    {
      do {
        okay = fread(&el, sizeof(struct Elemento), 1, pf);
        if (okay)
          printf("Matricola: %ld\tMaterie: %d\n", el.matricola, el.materie);
      } while(okay);
```

```
struct
```

```
        fclose(pf);
      }
    else
      exit(1);
}
```

□

Esercizio 4.5 ()

Scrittura e successiva lettura di strutture su file mediante funzione.

Risoluzione

```
#include <sli
```

```
      fclose(pf);
     }
    else
      exit(1);
}

void Leggi(char *nomefile)
{
  FILE            *pf;
  struct Elemento el;

  pf = fopen(nomefile, "r");
  if (pf)
    {
     while (fread(&el, sizeof(struct elemento), 1, pf) != 0)
         printf("Matricola: %ld\tMaterie: %d\n", el.matricola, el.materie);
     fclose(pf);
    }
  else
    {
      printf("Errore durante l'apertura del file");
      exit(1);
    }
}
```

□

4.2 `fseek()`

La funzione `fseek()` il cui prototipo è:

```
int fseek(FILE *fp, long spiazzamento, int da_dove);
```

setta l'indicatore di posizione del file per il flusso puntato da `fp`. La nuova posizione, misurata in bytes, è ottenuta aggiungendo `spiazzamento` bytes alla posizione specificata da `da_dove`. Se `da_dove` è settato a `SEEK_SET`, `SEEK_CUR`, o `SEEK_END`, lo spiazzamento è relativo rispettivamente all'inizio del file, alla posizione attuale dell'indicatore, o alla fine del file.

4.3 `ftell()`

La funzione `ftell()` il cui prototipo è:

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

restituisce la posizione corrente del file per il flusso puntato da `fp`.

Esercizio 4.6 ()

```
{
  FILE *pf;
  char nome_file[128];

  printf("Inserisci il nome di un file (percorso completo): ");
  scanf("%s", nome_file);

  pf = fopen(nome_file, "r");
  if (pf)
    {
      fseek(pf, SEEK_END);
      printf("%sè lungo %ld bytes", nome_file, ftell(pf));
      fclose(pf);
    }
  else
    {
      printf("%s non esiste!", nome_file);
    }
}
```

□