Gestione File di testo

In primo luogo diamo alcune nozioni di base su come è gestito l'I/O in ANSI C.

Osserviamo subito che in ANSI C non sono direttamente definite delle istruzioni per l' Input/Output (I/O). Esistono, tuttavia, delle funzioni definite nella libreria standard e dichiarate nel header file stdio.h che permettono tali operazioni.

E' importante sottolineare che tali funzioni consentono la lettura/scrittura in modo indipendente dalle caratteristiche proprie dei dispositivi di Input/Output. In altre parole, una stessa funzioni di I/O può essere utilizzata, ad esempio, sia per leggere un valore dalla tastiera sia per leggere un valore da un dispositivo di memoria di massa (file presente sul disco fisso, ad esempio). Lo stesso vale, ovviamente, per le funzioni di scrittura: la stessa funzione può essere utilizzata sia per la visualizzazione sullo schermo sia per scrivere un valore su un disco o una stampante. Tale caratteristica nasce dal modo in cui è gestito il sistema di I/O in ansi C. Tale sistema, infatti, è caratterizzato da un' *interfaccia* indipendente dal dispositivo effettivo che permette lo scambio di informazioni tra il programma e il dispositivo. Tale interfaccia è chiamata flusso (o stream).

I programmi C, quindi, leggono dati memorizzati in un dispositivo fisico, o scrivono dati su un dispositivo fisico, attraverso i flussi i quali sono sequenze di caratteri o byte(vedi Figura 12).

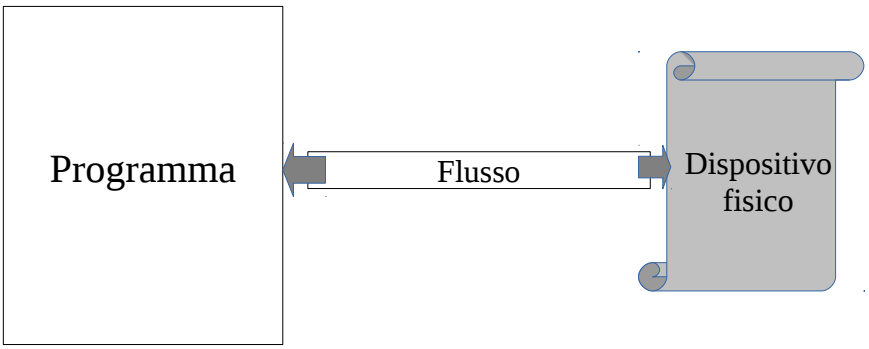

Figura 12: I programmi C accedono ai dati presenti su dispositivi fisici tramite i flussi (o stream)

La proprietà fondamentale di tale gestione dell'I/O è che tutti i flussi si comportano nella stessa maniera e, quindi, possono essere gestiti nella stessa maniera. Per tale motivo, siccome il sistema di I/O in C associa ad ogni dispositivo fisico un flusso, le stesse funzioni possono essere utilizzate per differenti dispositivi fisici (stampanti, file presente su una memorie di massa, schermo, etc...).

Una prima distinzione importante è che esistono due tipi di flussi:

- flussi di testo e
- flussi binari.

Un flusso di testo è una sequenza di caratteri. Tale sequenza è generalmente suddivisa in linee. Ciascuna linea è separata dalle altre in quanto terminate da un carattere speciale detto *newline*.

Un flusso binario è, invece, formato da una sequenza di byte. Si ha, inoltre, una corrispondenza uno ad uno con i byte presenti sul dispositivo fisico.

Andiamo ora, più nello specifico, a vedere come associare un flusso ad un dispositivo fisico.

Associazione tra flussi e dispositivi fisici.

Per associare un flusso a un dispositivo fisico (ad esempio, un file sul disco fisso, il monitor, etc...) è necessario un’operazione di apertura. Una volta associato un flusso ad un dispositivo fisico è possibile scambiare informazioni tra il dispositivo e il programma. Per eliminare l’associazione tra flusso ed il dispositivo è necessaria un’operazione di chiusura.

Quindi le prime due operazioni di base sono:

- **apertura** (associazione del flusso con il dispositivo)
- **chiusura** (eliminazione dell'associazione una volta terminate le operazioni sul dispositivo)

L'operazione di apertura può avvenire secondo diverse modalità. Possiamo distinguere, fondamentalmente, due diverse modalità:

- Apertura **in lettura**. In questo caso andiamo a leggere dal dispositivo fisico.
- Apertura **in scrittura**. In questo caso andiamo a scrivere sul dispositivo fisico.

In ANSI C la creazione di un flusso avviene tramite la creazione (tramite memoria dinamica) di un particolare record (una struct) chiamato FILE contenente, fondamentalmente, i seguenti campi:

- Modalità di utilizzo del dispositivo (lettura o scrittura, ad esempio);
- Posizione corrente su dispositivo (indicante il prossimo byte o carattere da leggere o scrivere sul dispositivo);
- Un indicatore di errore di lettura/scrittura;
- un indicatore di end-of-file, indicante il raggiungimento della fine del file.

Per poter aprire un flusso, quindi, è necessario usare delle variabili di tipo puntatore a FILE, dichiarate nel seguente modo:

```
FILE  *pf;
```

L'associazione di un flusso con un dispositivo fisico avviene tramite la funzione fopen che restituisce l'indirizzo di un nuovo oggetto FILE.

Tale funzione ha il seguente prototipo:

```
FILE *fopen (char nomeDispositivo , char *modo ) ;
```

dove 'nomeDispositivo' è una stringa di caratteri indicante il nome del dispositivo da aprire, ad esempio il nome di un file sul disco fisico. L'argomento 'modo' è una stringa che indica il modo in cui il file deve essere aperto.

Nel caso in cui si verifica un errore in apertura del file, la funzione fopen() restituisce un puntatore nullo (cioè NULL).

Ad esempio, per aprire in scrittura il file “prova.txt” scriveremo:

```
FILE *puntFile=fopen(“prova.txt”,”w”);
if ( puntFile = = NULL)
      {
         printf ( " Impossibile aprire  il   file " );
         }
```

Si osservi che, in generale, prima di accedere ad un dispositivo è necessario assicurarsi che la chiamata della funzione fopen() sia stata eseguita con successo, cioè che non abbia restituito NULL.

Una volta eseguite le operazioni desiderate sul dispositivo (un file del disco fisso, ad esempio) si deve chiudere lo stream. Tale chiusura avviene in ANSI C tramite la funzione fclose, la quale ha il seguente prototipo:

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

dove puntFile mantiene il valore restituto da una chiama della funzione fopen().

Un altra funzione importante è feof(). La funzione feof() restituisce un valore maggiore di zero (quindi un valore booleno “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 *puntFile);
```

Analogamente alla lettura e scrittura dallo standard input/output (ad esempio tastiera e schermo), sono definite le funzioni fscanf() e fprintf(). Tali funzioni sono 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, ...);
```

```
int fprintff(FILE *pf, const char stringa_di_controllo, ...);
```

Alcune altre funzioni per la lettura/scrittura su dispositivi.

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

```
Restituisce come unsigned char (covertito ad int) il successivo carattere
presente sullo stream, oppure EOF (Fine di File) se è raggiunta la fine del
file.
```

```
int fputc(int c, FILE *pf);
```

```
Scrive il carattere c (convertito ad unsigned char) sullo stream pf,
restituisce il carattere scritto oppure oppure EOF (Fine di File) in caso
di errore.
```

```
char *fgets(char *s, int n, FILE *pf);
```

```
Legge dalla stream pf al piu' i successivi n-1 caratteri ponendoli nel
vettore di caratteri s. Si blocca prima se incontra un newline '\n'. Il
newline viene incluso nel vettore s. Il vettore s termina con il carattere
di fine stringa '\0'. Resituisce s, oppure NULL se incontra la fine del
file o se c'e' un errore.
```

```
int fputs(const char *s, FILE *pf);
```

```
Scrive la stringa s sullo stream pf. Restituisce un valore positivo se tale
operazione va a buon fine, oppure EOF se c'e' un errore.
```

Esempio 1

Come primo esempio consideriamo un programma per la lettura di un file e la sua visualizzazione sullo schermo:

```
#include <stdio.h>
#define MAXLEN 50
void stampafile(char *);
int main() {
    char s[MAXLEN];
    printf("Dammi il nome del  file:\n");
    scanf("%s ",s);
    stampafile(s);
```

```
    return 0;
}

void stampafile(char *s) {
    FILE *fp;
    char c;
    fp=fopen(s,"r");
    if (fp!=NULL) {
           for (c=fgetc(fp); c != EOF ; c=fgetc(fp))
                          printf("%c",c);
           fclose(fp);
           }
        else printf("Il  FILE %s non si riesce ad aprire\n",s);
     }
```

Esempio 2

Il successivo esempio permette di copiare un file in un altro:

```
#include <stdio.h>
#define MAXLEN 50
void copiafile(char *,char *);
int main() {
    char s1[MAXLEN],s2[MAXLEN];
    printf("Dammi i nomi dei due file:\n");
    scanf("%s %s",s1,s2);
    copiafile(s1,s2);
    return 0;
}

void copiafile(char *s1,char *s2) {
    FILE *fp1,*fp2;
    char c;
```

```
    fp1=fopen(s1,"r");
    if (fp1!=NULL) {
       fp2=fopen(s2,"w");
       if (fp2!=NULL) {
                   for (c=fgetc(fp1); c != EOF ; c=fgetc(fp1))
                                       fprintf(fp2,"%c",c);
                   fclose(fp2);
                   }
        else printf("Il seconfo FILE %s non si riesce ad aprire\n",s2);
       fclose(fp1);
       }
     else printf("IL PRIMO FILE %s NON SI RIESCE AD APRIRE\n",s1);

     }
```

Esempio 3

Il successivo esempio mostra come leggere da un file di testo “parola per parola” e scrivere il tutto in un altro file in ordine inverso:

```
#include <stdio.h>
#define MAXLEN 50
#define MAXPAROLE 20
void copiaStringa(char *x,char *y,int n);
int main(){
 FILE *fp,*fp2; int i,j;
 char stringa[MAXLEN];
 char parole[MAXPAROLE][MAXLEN];
 fp=fopen("prova2.txt","r");
 if (fp!=NULL) {
               /* while (fscanf("%s",stringa)>0) */
               for (i=0;fscanf(fp,"%s",stringa)>0; ++i) {
                   copiaStringa(parole[i],stringa,MAXLEN);
                   }
```

```
                fp2=fopen("prova2_out.txt","w");
                if (fp2!=NULL) {
                       for (j=i-1; j>=0; --j)
                           fprintf(fp2,"%s ",parole[j]);
                       fclose(fp2);}
                else printf("Il file %s non si puo'
aprire\n","prova2_out.txt");
               fclose(fp);
               }
  else printf("Il file %s non esiste\n","prova2.txt");
 return 0;
}
void copiaStringa(char *x,char *y,int n) {
     int i=0;
     while (y[i]!= '\0' && i<n){
         x[i]=y[i];
           ++i;
           }
     x[i]='\0';    }
```

Esempio 4

L'esempio successivo mostra come leggere da un file di testo opportunamente organizzato una matrice M di interi (e come scrivere tale matrice in un file seguendo la stessa organizzazione). Si supponga che un file, ad esempio “matrice.txt” sia così organizzato: sulla prima riga sono presenti due numeri m ed n, indicati il numero di righe e colonne della matrice M, rispettivamente. Sulle successive m righe ci sono i valori delle delle m righe della matrice M separati da uno o più spazi. Sulla prima riga la prima riga della matrice M, sulla seconda riga la seconda riga della matrice M, e così via.

Ad esempio:

```
3 4
14    5     67    -21
23    -5    11    10
-1    3     12    18
```

Allora una possibile funzione C per leggere tale file è la seguente:

```
void leggiMatrice( FILE *fp, int M[][MAXC], int *p_m, int *p_n) {
      int i,j;
      fscanf(fp,"%d %d", p_m,p_n);
      for (i=0;i<(*p_m);++i)
            for (j=0;j<(*p_n);++j)
                  fscanf(fp,"%d",&M[i][j]);
}
void scriviMatrice( FILE *fp, int M[][MAXC], int m, int n) {
      int i,j;
      fprintf(fp,"%d %d\n", m, n);
      for (i=0;i<m;++i) {
            for (j=0;j<n;++j)
                  fprintf(fp,"%d ",M[i][j]);
            fprintf(fp,"\n"); /*Scritta una riga si va a capo!*/
      }
}
```

Tale funzione può essere utilizzata in un main del tipo come riportato nell'esempio successivo:

```
#include <stdio.h>
#define MAXR 100
#define MAXC 100
#define MAXS 50
int main() {
      FILE *fp1,*fp2=NULL;
      int M[MAXR][MAXC];
      int m,n;
      char nomeFileLettura[MAXS],nomeFileScrittura[MAXS];
      printf("Dammi il nome del file da cui leggere:");
      scanf("%s",nomeFileLettura);
      fp1=fopen(nomeFileLettura,"r");
      if (fp1!=NULL) {
```

```
            leggiMatrice(fp1,M,&m,&n);
            printf("Dammi il nome del file in cui scrivere:");
            scanf("%s",nomeFileScrittura);
            fp2=fopen(nomeFileScrittura,"w");
            if (fp2!=NULL) {
                  scriviMatrice(fp2,M,m,n);
                  fclose(fp2);
            }
            fclose(fp1);
      }
}
```

Si noti che a volte può essere utile leggere una sequenza di caratteri da file (ad esempio utilizzando fgetc in un ciclo terminate quando si incontra un particolare carattere oppure, sempre come esempio, fgets leggendo una rigo alla volta da un file), mettere tale sequenza in un stringa (vettore di caratteri con il carattere di fine stringa) ed operare, infine, su tale stringa per estrarre le informazioni corrette. Per estrarre tali informazioni si usa la seguente funzione (dichiarata in <stdio.h>):

int sscanf(char *s, const char *format, …);

La quale equivale a scanf ad eccezione del fatto che l'input viene prelevato dalla stringa s.

Esercizi

Esercizio 1.

Sia dato un file di testo *agenda.txt* contenente gli appuntamenti di una persona ordinati per data e orario. Le informazioni sono così strutturate: Ciascun rigo contiene una data (gg/mm) seguita dal numero *n* di appuntamenti per quella data. Seguono esattamente n coppie ognuna contenente un orario e una descrizione (in forma di stringa senza spazi). Scrivere una programma che dati in input due interi (rappresentati la data) e due interi (rappresentanti un orario iniziale e uno finale) restituisca in output la descrizione degli appuntamenti per quella data nell'intervallo di tempo specificato.

Esempio:

`agenda.txt`

`12/05 3 15 Riunione 16 Medico<eoln>`
`15/06 2 9 Lezione 14 Pranzo<eoln>`
`18/06 1 15 Riunione<eoln>`
`18/09 2 9 Lezione 14 Pranzo<eoln><eof>`
	`Input:`
	`(18,09) (12,20)`
	`Output``:`
	`14 Pranzo`

Esercizio 2.

In un file di testo "parole.txt" ogni rigo comincia con un numero che indica quante stringhe, separate da uno o più spazi, sono presenti nel rigo stesso. Realizzare un programma che scriva in un altro file di testo solo le stringhe che **non** si ripetono una o più volte nel file "parole.txt". Esempio:

Input:

4 mare palla pallone rete

5 marina sabbia marina mare sabbia

7 fiori petali fiori penne fiori pesci palla

Output: pallone rete petali penne pesci

Esercizio 3.

Sia dato un file testo di nome “voli.txt” contenente informazioni su alcuni voli. Per ogni volo l’informazione è così strutturata. Il primo rigo contiene il codice del volo, seguita dal carattere ‘:’ poi la città di partenza seguita dal carattere ‘;’ e dalla città di arrivo. Il secondo rigo contiene i giorni in cui il volo viene effettuato (Lunedì=1, Martedì=2, etc... 0 se nel giorno non c’e’ il volo). Il terzo, e ultimo, rigo contiene l’ora di partenza seguita dall’ora di arrivo.

Scrivere un programma che legga da standard input una città di partenza, una città di arrivo e un giorno della settimana e mostri a video il codice e gli orari dei voli corrispondenti.

Esempio del file “voli.txt”	Esempio:

`XG01102:Napoli;Barcellona`
`1 2 3 0 0 6 7`
`12:30 14:05`
`AZ07339:Napoli;Parigi`
`0 2 0 4 0 6 0`
`11:50 13:15`
	`citta’ di partenza = Napoli`
	`citta’ di arrivo = Parigi`
	`giorno (1,..,7) = 2`
	
	L’output sarà:
	`volo: AZ07339`
	`orario di partenza: 11:50`
	`orario di arrivo: 13:15`