FILE SEQUENZIALI E AD ACCESSO DIRETTO

L'input-output astratto (stream)

Le periferiche disponibili in un sistema di elaborazione sono, dal punto di vista hardware, anche molto diverse fra di loro anche se dal punto di vista dell'utente le funzioni che svolgono possono essere assimilabili.

Si pensi, per esempio, alle differenze sostanziali fra una stampante ad aghi e ad una laser: dal punto di vista dell'utente si tratta in tutte e due i casi di una stampante (una periferica che produce un output su supporto cartaceo), dal punto di vista hardware si tratta di due cose distinte almeno quanto lo potrebbero essere, per esempio, una stampante ad aghi e un monitor.

Il sistema operativo fornisce un'interfaccia ad alto livello verso l'hardware: le periferiche sono mappate in memoria, è utilizzata cioè in pratica una parte della memoria centrale (il *buffer*) come deposito temporaneo dei dati da e verso le periferiche.

In questo modo, per esempio, le operazioni di input possono essere effettuate sempre allo stesso modo a prescindere dalla periferica: sarà il sistema che si occuperà della gestione della specificità dell'hardware.

Il sistema di I/O fornisce il concetto astratto di **canale** (*Flusso* o S*tream*).

Con tale termine si intende un dispositivo logico indipendente dalla periferica fisica: chi scrive il programma si dovrà occupare dei dati che transitano per il canale prescindendo dalle specifiche del dispositivo fisico che sta usando (un lettore di dischi magnetici, una stampante).

Il termine **file** si riferisce invece ad una astrazione che è applicata a qualsiasi tipo di dispositivo fisico.

In poche parole, si potrebbe affermare che il file rappresenta il modo attraverso il quale l'utilizzatore vede sistemati i dati sul dispositivo di I/O, e che il canale è il modo con cui i dati sono accessibili per l'utilizzazione.

Un canale è associato ad un file per mezzo di una **open** (apertura del file), a questo punto i dati presenti nel file sono accessibili per l'utilizzo.

L'associazione fra canale e file è eliminata per mezzo di una **close** che interrompe le comunicazioni. Se il canale è stato aperto per operazioni di output, la chiusura consente di scrivere sul dispositivo fisico i dati ancora presenti nel buffer (lo *svuotamento del canale*).
In questo modo si evita che una parte dei dati possa non essere registrata.

L'associazione di canali alla tastiera per l'input e al video per l'output è curata in automatico dal sistema operativo per consentire il dialogo con il sistema.

La tastiera e il video sono cioè le *periferiche di default*: il sistema è già connesso con esse. Per quanto riguarda invece le comunicazioni con altre periferiche è necessario esplicitare l'associazione di canali per tali comunicazioni.

Dati su memorie di massa (file sequenziali)

Le strutture dati, prevedono la registrazione degli elementi su locazioni di memoria centrale.
La memoria centrale, in ragione delle sue caratteristiche, si presta a due tipi di utilizzo:

- Per la sua **limitatezza** è opportuno utilizzarla per la conservazione dei dati strettamente indispensabili per l'elaborazione in corso.
- Per la sua **volatilità** consente di conservare i dati limitatamente al solo tempo di esecuzione di un programma.

Se si vogliono conservare grandi quantità di dati e utilizzarli in tempi diversi, è necessario affidarsi a supporti permanenti che non hanno le limitazioni della memoria centrale: sono quelli che comunemente sono chiamate memorie di massa (*link a caratteristiche delle memorie di massa*).

Nell'esempio successivo si gestiranno una serie di libri, conservando in un file su memoria di massa i dati sugli stessi. I dati sui libri saranno registrati uno di seguito all'altro: quando si tratterà di rileggerli, l'ordine di reperimento delle informazioni sarà lo stesso di quello che si è utilizzato per la scrittura. I dati saranno quindi elaborati in maniera sequenziale.

```
/* Gestione File sequenziale */
#include <stdio.h>

/* Definizione record */

typedef struct {
  char titolo[50];
  char autore[20];
  char editore[20];
  long int prezzo;
}libro;

/* Funzioni per la gestione del file */

void inserisci();
void esamina();

main(){
  int scelta;
  for(;;){

    /* Menu operazioni disponibili */

    printf("\nGestione di un file contenente libri\n");
    printf("\n1) Inserimento di un libro nel file");
    printf("\n2) Esame dei libri contenuti nel file");
    printf("\n0) Fine elebaorazione\n");
    printf("\nScelta operazione (1..2,0) ");
    scanf("%d",&scelta);
    if(!scelta)
      break;

    /* Richiama funzione scelta */

    switch(scelta){
      case 1:
        inserisci();
        break;
```

```
        case 2:
          esamina();
      }
     }
     return 0;
    }

   /* Inserimento di un libro nel file */

   void inserisci(){



     /* viene definito il buffer che conterrà il libro da registrare nel file.*/

     libro buflib;

     /* viene dichiarato un puntatore (file pointer) alla struttura FILE.
        Tale struttura, definita in stdio.h, in accordo con quanto espresso prima,
        rappresenterà il canale associato al file nel quale saranno conservati i dati sui libri.*/

    FILE *fp;

    /* viene effettuata una chiamata alla funzione fopen.
       Tale chiamata ritorna un puntatore al tipo FILE.
       I parametri da passare alla funzione richiedono di specificare una stringa contenente
       il nome con il quale il file è registrato nella memoria di massa e una stringa specificante
       la modalità di apertura del file.
       In questo caso il file è aperto in modalità append:
       i record saranno registrati di seguito a quelli già presenti nel file. Se il file non esiste verrà
creato. */


     fp=fopen("LibriSeq.dat","a");




     if(fp==NULL)                          /*4*/
       return;
     printf("\nInserimento di un libro");
     fflush(stdin);
     printf("\nTitolo : ");
     gets(buflib.titolo);
     printf("Autore : ");
     gets(buflib.autore);
     printf("Editore : ");
     gets(buflib.editore);
     printf("Prezzo : ");
     scanf("%ld",&buflib.prezzo);

     /* conservazione nel file */

     fputs(buflib.titolo,fp);              /*5*/
     fprintf(fp,"\n");
```

```
  fputs(buflib.autore,fp);
  fprintf(fp,"\n");
  fputs(buflib.editore,fp);
  fprintf(fp,"\n");
  fprintf(fp,"%ld\n",buflib.prezzo);
  fclose(fp);                              /*6*/
}

/* Scansione sequenziale del file dei libri */

void esamina(){
  libro buflib;
  FILE *fp;
  fp=fopen("LibriSeq.dat","r");            /*7*/
  if(fp==NULL)
    return;
  for(;;){
    if (fgets(buflib.titolo,50,fp)==NULL)  /*8*/
      break;
    fgets(buflib.autore,20,fp);            /*9*/
    fgets(buflib.editore,20,fp);
    fscanf(fp,"%ld",&buflib.prezzo);
    fgetc(fp);                             /*10*/
    puts(buflib.titolo);
    puts(buflib.autore);
    puts(buflib.editore);
    printf("%ld",buflib.prezzo);
    fflush(stdin);
    while(!getchar()=='\n');
  }
  fclose(fp);
}
```

Il puntatore ritornato dalla fopen può essere NULL se il sistema non ha potuto generare, per un motivo qualsiasi, il file richiesto (per esempio se il dischetto è protetto da scrittura o se non c'è più spazio). Tale eventualità è testata nella 4.

A cominciare da 5 per scrivere i dati sul file, così come messo in evidenza prima, vengono utilizzate le funzioni fputs e fprintf formalmente uguali alle puts e printf più volte utilizzate. La funzione fputs, a differenza della puts, richiede di specificare oltre alla stringa da scrivere, la specifica del canale attraverso il quale effettuare l'operazione. Una osservazione simile vale per la fprintf, solo che stavolta il canale va specificato come primo parametro. Ogni registrazione di un singolo dato è seguita dalla registrazione di un carattere *newline* al fine di un più semplice reperimento dei singoli dati in fase di lettura. Per maggiori chiarimenti si legga quanto osservato nella funzione di lettura dal file.

La funzione fclose della 6 si occupa di interrompere le comunicazioni con il file.

Per poter accedere ai dati contenuti nel file, è necessario aprirlo in modalità *read*. Questo è ciò di cui si occupa l'istruzione contenuta nella riga 7.

La funzione esamina legge tutti i record registrati nel file e li visualizza sul video. A tal fine usa, come specificato in 8 e 9 le funzioni fgets e fscanf. La funzione fgets necessita, oltre che della specifica della stringa dove depositare la lettura, della quantità massima di caratteri da leggere e del canale attraverso il quale effettuare la lettura. È opportuno notare che la funzione legge dal file fino al numero di caratteri specificato o al *newline* se questo viene incontrato prima. Questo è il motivo dell'inserimento di tale carattere come delimitatore dei singoli dati: non si può infatti conoscere

l’esatta lunghezza del dato contenuto nel campo e così si utilizza il *newline* come delimitatore. La funzione fgetc della 10 legge l’ultimo *newline* inserito dopo l’ultima fprintf in sede di scrittura.

L’elaborazione sequenziale di un file prevede la lettura di tutte le registrazioni contenute in esso così come sono state registrate. Per verificare l’avvenuta lettura di tutte le registrazioni contenute nel file, si è inserito il controllo specificato in 8. Se i libri registrati sono stati tutti letti, un ulteriore tentativo di lettura di una stringa ritorna un puntatore NULL.

File ad accesso casuale o diretto

Se il supporto sul quale sono registrati i dati è gestito da un dispositivo che lo consente, è possibile accedere ad una registrazione qualsiasi contenuta nel file, specificando la sua posizione relativa all'interno del file stesso. La logica di gestione di un file ad accesso casuale somiglia a quella conosciuta della gestione di una tabella. Anche in quel caso i records erano accessibili specificando la loro posizione relativa. Spingendo la similitudine con gli argomenti trattati in precedenza si può dire che i records in un file sequenziale sono elaborati come gli elementi di una coda, in un file ad accesso casuale come una tabella.

```
/*
  Gestione file di libri con elaborazione sequenziale
  ed accesso diretto
*/
#include <stdio.h>

/* Definizione del record */

typedef struct {
  char titolo[50];
  char autore[20];
  char editore[20];
  long int prezzo;
}libro;
...
/* Inserimento di un libro nel file */

void inserisci(){                          /*1*/
  FILE *fp;
  libro buflib;
  fp=fopen("DatiLib.dat","a");                  /*2*/
  if(fp==NULL)
    return;
  printf("\nInserimento di un libro");
  fflush(stdin);
  printf("\nTitolo : ");
  gets(buflib.titolo);
  printf("Autore : ");
  gets(buflib.autore);
  printf("Editore : ");
  gets(buflib.editore);
  printf("Prezzo :");
  scanf("%ld",&buflib.prezzo);
  fwrite(&buflib,sizeof(libro),1,fp);           /*3*/
  fclose(fp);
}

/* Estrazione di un libro dal file */

void estrai(){                             /*4*/
  FILE *fp;
  libro buflib;
  int quale;
  long dove;
  fp=fopen("DatiLib.dat","r");                  /*5*/
  if(fp==NULL)
    return;
```

```
  printf("\nEstrazione di un libro dal file");
  printf("\nQuale libro :");
  scanf("%d",&quale);                          /*6*/
  dove=(long) sizeof(libro)*(quale-1);         /*7*/
  fseek(fp,dove,SEEK_SET);                     /*8*/
  if(!fread(&buflib,sizeof(libro),1,fp)){      /*9*/
    printf("\nLibro inesistente");
    return;
  }
  puts(buflib.titolo);
  puts(buflib.autore);
  puts(buflib.editore);
  printf("%ld\n",buflib.prezzo);
  fflush(stdin);
  while(!getchar()=='\n');
  fclose(fp);
}

/* Scansione sequenziale del file */

void esamina(){                                /*10*/
  FILE *fp;
  libro buflib;
  fp=fopen("DatiLib.dat","r");
  if(fp==NULL)
    return;
  printf("\nLibri conservati nel file\n");
  fread(&buflib,sizeof(libro),1,fp);           /*11*/
  while(!feof(fp)){                            /*12*/
    puts(buflib.titolo);
    puts(buflib.autore);
    puts(buflib.editore);
    printf("%ld\n",buflib.prezzo);
    fflush(stdin);
    while(!getchar()=='\n');
    fread(&buflib,sizeof(libro),1,fp);         /*11*/
  }
  fclose(fp);
}

/* Modifica dati di un libro esistente */

void modifica(){                               /*12*/
  FILE *fp;
  libro buflib;
  int quale;
  long dove;

  fp=fopen("DatiLib.dat","r+");                /*13*/
  if(fp==NULL)
    return;

  printf("\nModifica dati libro dal file");    /*14*/
  printf("\nQuale libro :");
  scanf("%d",&quale);
  dove=(long) sizeof(libro)*(quale-1);
  fseek(fp,dove,SEEK_SET);
```

```
    if(!fread(&buflib,sizeof(libro),1,fp)){
      printf("\nLibro inesistente");
      return;
    }
    puts(buflib.titolo);
    puts(buflib.autore);
    puts(buflib.editore);
    printf("%ld\n",buflib.prezzo);

    /* Nuovi dati */

    printf("\nNuovi dati libro");                /*15*/
    fflush(stdin);
    printf("\nTitolo  : ");
    gets(buflib.titolo);
    printf("Autore  : ");
    gets(buflib.autore);
    printf("Editore : ");
    gets(buflib.editore);
    printf("Prezzo  :");
    scanf("%ld",&buflib.prezzo);

    fseek(fp,dove,SEEK_SET);                     /*16*/
    fwrite(&buflib,sizeof(libro),1,fp);          /*17*/

    fclose(fp);
  }
```

La funzione inserisci definita in 1 si occupa, allo stesso modo dell'equivalente nell'esempio del file sequenziale, della conservazione di un libro nel file. Formalmente è molto simile all'altra vista in precedenza. Anche in questo caso c'è un buffer per la conservazione temporanea del record da registrare e un puntatore alla struttura FILE. Anche l'apertura del file viene effettuata, come si legge in 2, per mezzo di una chiamata alla funzione fopen.

Per quanto riguarda la scrittura dei dati sul file, in questo caso, viene utilizzata nella 3 la funzione fwrite che si occupa di scrivere un blocco di byte. Tale funzione richiede come parametri un puntatore all'area da cui prelevare i dati da scrivere (&buflib), il numero di byte che devono essere scritti (sizeof(libro)), quanti blocchi di quella dimensione scrivere (1), il canale da utilizzare (fp).

La funzione estrai della 4, dopo aver aperto in 5 il file similmente alla equivalente nel file sequenziale, si occupa di rintracciare una specifica registrazione all'interno del file, una volta conosciuta la dimensione comune dei records conservati nel file e la posizione del record interessato. Il metodo utilizzato, per il rintracciamento del record interessato, è lo stesso di quello utilizzato per il rintracciamento di un elemento in una struttura sequenziale (indirizzo base, scostamento da effettuare): nella 6 viene richiesto di specificare la posizione del record da leggere, nella 7 tale posizione relativa viene trasformata in indirizzo relativo. Si noti che l'espressione qui utilizzata è l'equivalente della seconda parte di $IND(x_i)=IND_b+(i-1)l$, infatti qui la dimensione comune degli elementi è rappresentata da sizeof(libro). L'espressione per esteso è scritta nella 8, dove la funzione fseek ci posiziona sul record interessato. La funzione richiede infatti come parametri oltre che il puntatore al file (fp) l'indirizzo base da cui partire (SEEK_SET è il simbolo utilizzato per indicare l'inizio del file ed è definito in stdio.h), lo scostamento relativo calcolato in precedenza. Si noti che è semplicemente un modo diverso di scrivere l'espressione ricordata prima.

Effettuato il posizionamento, la funzione fread della 9 si occupa di depositare nel buffer predisposto (l'area di memoria associata a &buflib) il record letto. La funzione richiede gli stessi parametri della fwrite utilizzata in 3. La funzione ritorna un puntatore all'area utilizzata per conservare il record

letto (lo stesso valore di &buflib). Se l'operazione di lettura non ha avuto esito positivo, per esempio perché non esiste un record registrato all'indirizzo specificato, il puntatore assume valore NULL: tale condizione è testata appunto nella 9.

I record conservati in un file ad accesso casuale possono essere elaborati anche sequenzialmente ed è di questo tipo di elaborazione che si occupa la funzione esamina della 10. Anche in questo caso, dopo aver effettuato le solite operazioni, si procede alla lettura di tutti i record contenuti nel file nello stesso ordine in cui sono registrati: a tale fine è utilizzata la funzione fread delle 11. La prima volta del suo utilizzo, poiché non è stata utilizzata la fseek, viene letto il primo record registrato. Ogni nuova chiamata alla fread ritorna in buflib il record successivo e in questo modo possono essere letti tutte le registrazioni contenute nel file. Non conoscendo la quantità di record registrati nel file, è necessario utilizzare la funzione feof. Tale funzione ritorna il valore di verità (1) se nell'ultima operazione di lettura, dal canale specificato come parametro (fp nell'esempio), si è raggiunti la fine del file.

La funzione modifica che comincia da 12, si occupa dell'update dei dati di un libro già registrato precedentemente nel file. Per buona parte è formalmente simile alla estrai della 4: anche in questo caso si tratta di posizionarsi su uno specifico record del file e leggerlo, solo che stavolta se ne vuole registrare una nuova versione modificata. Nella 13 si apre in modalità update (identificata dal parametro r+ della open) il file interessato. Il gruppo di istruzioni che comincia da 14 è lo stesso di quello che si ritrova nella estrai: si cerca il libro interessato dalla modifica, lo si carica in memoria e si visualizza l'attuale contenuto dei campi. A cominciare dalla 15 si ricevono i nuovi dati da registrare. Nella 16 ci si riposiziona nel punto da cui si è letto il record. Si sono già fatte notare le analogie con la tabella: anche qui per effettuare operazioni su singoli elementi deve essere specificata la relativa posizione. La 17, identica alla 3, si occupa di scrivere il record nella posizione prima specificata.