Capitolo 11 – Elaborazione di file

11.1 Introduzione

- I file
  - Possono essere creati, modificati, ed elaborati da programmi scritti in C
  - Sono utilizzati per la memorizzazione permanente dei dati
    - La memorizzazione di dati in variabili ed array è solo temporanea

11.2 La gerarchia dei dati

- Gerarchia dei dati:
  - Bit – il più piccolo
    - Valore `0` o `1`
  - Byte – 8 bits
    - Utilizzato per memorizzare un carattere
      - Cifre decimali, lettere, e simboli speciali
  - Campi – gruppo di caratteri
    - Esempio: your name
  - Record – gruppo di campi
    - Rappresentato da `struct` o `class`
    - Esempio: In a payroll system, a record for a particular employee that contained his/her identification number, name, address, etc.

11.2 La gerarchia dei dati

- Gerarchia dei dati:
  - File – gruppo di record
    - Esempio: payroll file
  - Database – gruppo di files

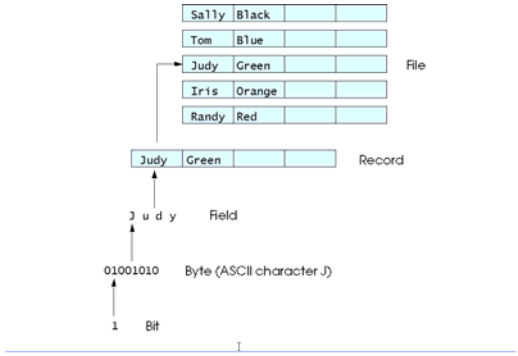

Fig. 11.1 The data hierarchy.

5

11.2 La gerarchia dei dati

- File
  - Gruppo di record correlati
  - Chiave del Record
    - Identifica un record per facilitare il ritrovamento di uno specifico record da un file
  - File sequenziale
    - I record sono tipicamente ordinati in base alla chiave

6

I file in C

- Il C, al contrario di altri linguaggi, offre soltanto alcune funzioni di base per operare sui file:
  - non esiste il concetto di record e di chiave di ricerca;
  - non esistono pertanto neppure le classiche funzioni di inserimento, modifica, cancellazione e ricerca di record all'interno di un file.
  - È consentito operare solamente sui file costituiti da sequenze lineari di byte, mediante funzioni per la creazione, rimozione di file, lettura e scrittura di byte da/su file

7

Apertura di file

- La prima operazione da eseguire prima di poter scrivere o leggere da file è l'apertura:

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

  - In stdio.h è contenuta la definizione del tipo derivato FILE
- La funzione fopen apre il file di nome ordini;
  - l'operazione di apertura serve ad associare la variabile puntatore fp, nota con il nome di *file pointer*, al file di nome `ordini`.
  - Dopo che il file è stato aperto, si dovrà utilizzare il file pointer fp per poter scrivere e leggere le informazioni memorizzate al suo interno.
  - Nel caso in cui si verifichi un errore, per esempio perché il file `ordini` non esiste, la funzione fopen ritorna un file pointer NULL.

8

Chiusura di file

- Dopo avere terminato le operazioni di lettura e/o scrittura è necessario eseguire l'operazione di chiusura del file:

```
fclose(fp);
```

  - La chiusura del file garantisce che tutti i dati scritti nel file ordini siano salvati su disco;
  - infatti molto spesso il sistema operativo, per ottimizzare le prestazioni del programma, ritarda le scritture sulla memoria di massa mantenendo le informazioni temporaneamente in memoria centrale.

Modi di apertura di un File

- Al momento dell'apertura è sempre necessario specificare, oltre al nome del file, anche il tipo di operazione che si desidera eseguire.
  - Nell'esempio precedente il parametro "r" stava a indicare che il file doveva essere aperto in lettura (*read*).
  - In generale, le possibili modalità di apertura di un file, specificate dal secondo parametro di fopen, sono le seguenti.

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

11.3 File e Stream

- Il C vede un file come un flusso (stream) sequenziale di byte
  - I file terminano con un marcatore di end-of-file
    - Oppure, terminano con uno specifico byte

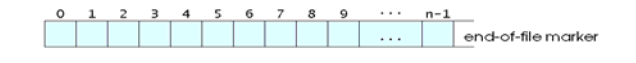

Fig. 11.2 C's view of a file of *n* bytes.

- Quando si apre un file si crea uno stream
  - Fornisce il canale di comunicazione tra file e programma
  - L'apertura di un file restituisce un puntatore ad una struttura `FILE`
    - Esempi:
    - `stdin` - standard input (keyboard)
    - `stdout` - standard output (screen)
    - `stderr` - standard error (screen)

Lettura e scrittura su file

- Funzioni di lettura/scrittura
  - `fgetc`
    - Legge un carattere da un file
    - Puntatore `FILE` come argomento
    - `fgetc( stdin )` è equivalente a `getchar()`
  - `fputc`
    - Scrive un carattere su un file
    - Puntatore `FILE` e un carattere come argomento
    - `fputc( 'a', stdout )` è equivalente a `putchar( 'a' )`
  - `fgets`
    - Legge una linea da un file
  - `fputs`
    - Scrive una linea su un file
  - `fscanf` / `fprintf`
    - Versioni per file equivalenti di `scanf` e `printf`

11.4 Creazione di un file ad accesso sequenziale

- Il C non impone una struttura file
  - Non esiste la nozione di record in un file
  - Il programmatore deve fornire la struttura
- Creare un file
  - `FILE *cfPtr;`
    - Crea un puntatore `FILE` chiamato `cfPtr`
  - `cfPtr = fopen(“clients.dat", “w”);`
    - La funzione `fopen` restituisce un puntatore `FILE` al file specificato
    - Ha due argomenti – il file da aprire e la modalità d’apertura
    - Se l’apertura fallisce viene restituito `NULL`

11.4 Creazione di un file ad accesso sequenziale

- `fprintf`
  - Utilizzato per stampare in un file
  - Come la `printf`, eccetto che il primo argomento è un puntatore `FILE` (puntatore al file in cui si vuole stampare)
- `feof(` *`FILE pointer`* `)`
  - Restituisce l'indicatore end-of-file
- `fclose(` *`FILE pointer`* `)`
  - Chiude il file specificato
  - Eseguito automaticamente quando termina il programma
  - È buona pratica chiudere il file esplicitamente

- Dettagli
  - I programmi possono operare su un file, su più file o su nessun file
  - Ogni file deve avere un nome unico ed il suo rispettivo puntatore

Outline

fig11_03.c (1 of 2)

```
1  /* Fig. 11.3: fig11_03.c
2     Create a sequential file */
3  #include <stdio.h>
4
5  int main()
6  {
7     int account;     /* account number */
8     char name[ 30 ]; /* account name */
9     double balance;  /* account balance */
10
11    FILE *cfPtr;     /* cfPtr = clients.dat file pointer */
12
13    /* fopen opens file. Exit program if unable to create file  */
14    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } /* end if */
17    else {
18       printf( "Enter the account, name, and balance.\n");
19       printf( "Enter EOF to end input.\n" );
20       printf( "? " );
21       scanf( "%d%s%lf", &account, name, &balance );
22
```

Outline

```
23       /* write account, name and balance into file with fprintf */
24       while ( !feof( stdin ) ) {
25          fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
26          printf( "? " );
27          scanf( "%d%s%lf", &account, name, &balance );
28       } /* end while */
29
30       fclose( cfPtr ); /* fclose closes file */
31    } /* end else */
32
33    return 0; /* indicates successful termination */
34
35 } /* end main */
```

fig11_03.c (2 of 2)

Program Output

```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

Computer system	Key combination
UNIX systems	*<return> <ctrl> d*
IBM PC and compatibles	*<ctrl> z*
Macintosh	*<ctrl> d*

Fig. 11.4 End-of-file key combinations for various popular computer systems.

11.5 Lettura da un file ad accesso sequenziale

- Lettura
  - Crea un puntatore `FILE`, collegarlo al file da leggere
    `cfPtr = fopen( “clients.dat", "r" );`
  - Utilizzare la `fscanf` per leggere dal file
    - Come la `scanf`, eccetto che il primo argomento è un puntatore `FILE`
      `fscanf( cfPtr, "%d%s%f", &accounnt, name, &balance );`
  - Dati letti dall’inizio alla fine
  - Punatatore di posizione del file
    - Indica il numero del prossimo byte da leggere/scivere
    - Non è veramente un puntatore, ma un valore intero (specifica la locazione in byte)
    - Detto anche byte offset
  - `rewind( cfPtr )`
    - Riposizionamento del punatatore posizione all’inizio del file (byte `0`)

Outline

fig11_07.c (1 of 2)

```
1  /* Fig. 11.7: fig11_07.c
2     Reading and printing a sequential file */
3  #include <stdio.h>
4
5  int main()
6  {
7     int account;     /* account number */
8     char name[ 30 ]; /* account name */
9     double balance;  /* account balance */
10
11    FILE *cfPtr;     /* cfPtr = clients.dat file pointer */
12
13    /* fopen opens file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } /* end if */
17    else { /* read account, name and balance from file */
18       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
19       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
20
21       /* while not end of file */
22       while ( !feof( cfPtr ) ) {
23          printf( "%-10d%-13s%7.2f\n", account, name, balance );
24          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
25       } /* end while */
26
```


Outline

fig11_07.c (2 of 2)

```
27       fclose( cfPtr ); /* fclose closes the file */
28    } /* end else */
29
30    return 0; /* indicates successful termination */
31
32 } /* end main */
```

```
Account   Name          Balance
100       Jones           24.98
200       Doe            345.67
300       White            0.00
400       Stone          -42.16
500       Rich           224.62
```

11.5 Lettura da un file ad accesso sequenziale

- File ad accesso sequenziale
  - Non può essere modificato senza il rischio di distruggere i dati
  - I campi possono variare in dimensione
    - Diverse rappresentazioni nei file
    - `1`, `34`, `-890` sono tutti `int`, ma hanno diverse dimensioni su disco

```
300 White 0.00 400 Jones 32.87   (old data in file)
```

If we want to change White's name to Worthington,

```
300 Worthington 0.00
```

```
300 White 0.00 400 Jones 32.87
```

Data gets overwritten

```
300 Worthington 0.00ones 32.87
```


Outline

fig11_08.c (1 of 5)

```
1  /* Fig. 11.8: fig11_08.c
2     Credit inquiry program */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int request;     /* request number */
9     int account;     /* account number */
10    double balance;  /* account balance */
11    char name[ 30 ]; /* account name */
12    FILE *cfPtr;     /* clients.dat file pointer */
13
14    /* fopen opens the file; exits program if file cannot be opened */
15    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
16       printf( "File could not be opened\n" );
17    } /* end if */
18    else {
19
20       /* display request options */
21       printf( "Enter request\n"
22               " 1 - List accounts with zero balances\n"
23               " 2 - List accounts with credit balances\n"
24               " 3 - List accounts with debit balances\n"
25               " 4 - End of run\n? " );
```

```
26       scanf( "%d", &request );
27
28       /* process user's request */
29       while ( request != 4 ) {
30
31          /* read account, name and balance from file */
32          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
33
34          switch ( request ) {
35
36             case 1:
37                printf( "\nAccounts with zero balances:\n");
38
39                /* read file contents (until eof) */
40                while ( !feof( cfPtr ) ) {
41
42                   if ( balance == 0 ) {
43                      printf( "%-10d%-13s%7.2f\n",
44                              account, name, balance );
45                   } /* end if */
46
47                   /* read account, name and balance from file */
48                   fscanf( cfPtr, "%d%s%lf",
49                           &account, name, &balance );
50                } /* end while */
51
```


21

fig11_08.c (2 of 5)

```
52                break;
53
54             case 2:
55                printf( "\nAccounts with credit balances:\n");
56
57                /* read file contents (until eof) */
58                while ( !feof( cfPtr ) ) {
59
60                   if ( balance < 0 ) {
61                      printf( "%-10d%-13s%7.2f\n",
62                              account, name, balance );
63                   } /* end if */
64
65                   /* read account, name and balance from file */
66                   fscanf( cfPtr, "%d%s%lf",
67                           &account, name, &balance );
68                } /* end while */
69
70                break;
71
72             case 3:
73                printf( "\nAccounts with debit balances:\n");
74
```


22

fig11_08.c (3 of 5)

```
75                /* read file contents (until eof) */
76                while ( !feof( cfPtr ) ) {
77
78                   if ( balance > 0 ) {
79                      printf( "%-10d%-13s%7.2f\n",
80                              account, name, balance );
81                   } /* end if */
82
83                   /* read account, name and balance from file */
84                   fscanf( cfPtr, "%d%s%lf",
85                           &account, name, &balance );
86                } /* end while */
87
88                break;
89
90             } /* end switch */
91
92             rewind( cfPtr ); /* return cfPtr to beginning of file */
93
94             printf( "\n? " );
95             scanf( "%d", &request );
96          } /* end while */
97
```


Outline

fig11_08.c (4 of 5)

```
98          printf( "End of run.\n" );
99          fclose( cfPtr ); /* fclose closes the file */
100      } /* end else */
101
102      return 0; /* indicates successful termination */
103
104   } /* end main */
```


Outline

fig11_08.c (5 of 5)

Program Output

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
300       White           0.00

? 2

Accounts with credit balances:
400       Stone         -42.16

? 3

Accounts with debit balances:
100       Jones          24.98
200       Doe           345.67
500       Rich          224.62

? 4
End of run.
```

Outline

Esercizi

- Scrivere un programma che legga e visualizzi il contenuto di un file ASCII, per esempio autoexec.bat.
- Scrivere un programma che apra un file e vi inserisca 80 caratteri.
- Scrivere un programma che apra un file sequenziale contenente del testo e conti il numero di righe.