File position pointer

- File position pointer
  - Indica la posizione da cui leggere/scrivere I prossimi byte
  - Non un puntatore, ma un valore intero (specifica locazioni di bytes)
  - E` anche chiamato byte offset
- `rewind( cfPtr )`
  - Riposiziona il file position pointer all'inizio del file (byte 0).

Struttura in memoria

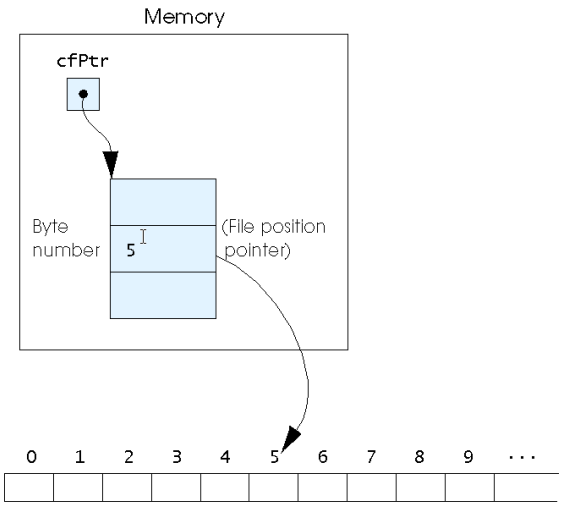


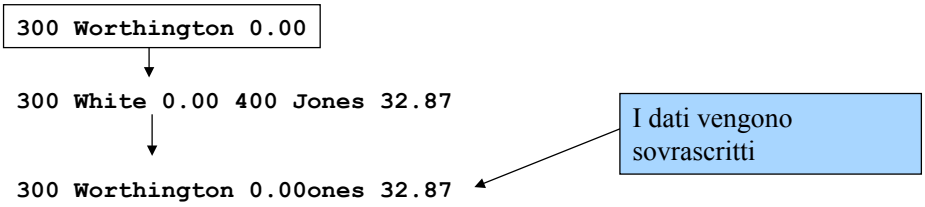
Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

Files ad accesso sequenziale:

- Se vengono modificati i dati possono essere distrutti.
- I campi possono variare in dimensione
  - (non sono la rappresentazione interna)
  - `1`, `34`, `-890` sono tutti interi `int`, ma possono avere differenti rappresentazioni.

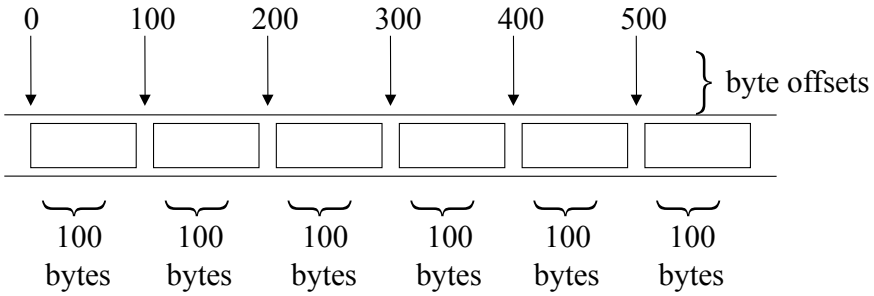
```
300 White 0.00 400 Jones 32.87    (vecchio valore)
```

Se vogliamo cambiarlo:



Files ad accesso diretto

- Files ad accesso diretto:
  - Accedono i record individualmente e direttamente
  - I dati possono essere inseriti senza distruggere altri dati
  - I dati precedenti possono essere modificati o cancellati.
- Realizzati usando record di lunghezza fissa
  - (I files sequenziali non hanno lunghezza fissa



Creare un file ad accesso diretto (binario)

- Dati:
  - Non sono formattati (sono solo bytes)
    - Tutti i dati dello stesso tipo usano la stessa memoria (**`int`**, per esempio)
    - Tutti i record dello stesso tipo hanno la stessa lunghezza
    - I dati sono “illeggibili” per l’utente

Posizionamento del puntatore di posizione all’interno del file

- `fseek`
  - Setta il file position pointer ad una certa posizione
  - `fseek(` *pointer, offset, symbolic_constant* `);`
    - *pointer* – puntatore al file
    - *offset* – file position pointer (0 e` la prima locazione del file)
    - *symbolic_constant* – specifica da dove partire
    - `SEEK_SET` – dall’inizio del file
    - `SEEK_CUR` – dalla corrente locazione
    - `SEEK_END` – alla fine del file

```c
/* Fig. 11.11: fig11_11.c
   Creating a randomly accessed file sequentially */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;           /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;        /* account balance */
}; /* end structure clientData */

int main()
{
   int i; /* counter */

   /* create clientData with no information */
   struct clientData blankClient = { 0, "", "", 0.0 };

   FILE *cfPtr; /* credit.dat file pointer */

   /* fopen opens the file; exits if file cannot be opened */
   if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
```



```c
   else {

      /* output 100 blank records to file */
      for ( i = 1; i <= 100; i++ ) {
         fwrite( &blankClient, sizeof( struct clientData ), 1, cfPtr );
      } /* end for */

      fclose ( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */
```

```c
/* Fig. 11.12: fig11_12.c
   Writing to a random access file */
#include <stdio.h>

/* clientData structure definition */
struct clientData {
   int acctNum;          /* account number */
   char lastName[ 15 ];  /* account last name */
   char firstName[ 10 ]; /* account first name */
   double balance;       /* account balance */
}; /* end structure clientData */

int main()
{
   FILE *cfPtr; /* credit.dat file pointer */

   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

   /* fopen opens the file; exits if file cannot be opened */
   if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
   else {

```



```c
      /* require user to specify account number */
      printf( "Enter account number"
              " ( 1 to 100, 0 to end input )\n? " );
      scanf( "%d", &client.acctNum );

      /* user enters information, which is copied into file */
      while ( client.acctNum != 0 ) {

         /* user enters last name, first name and balance */
         printf( "Enter lastname, firstname, balance\n? " );

         /* set record lastName, firstName and balance value */
         fscanf( stdin, "%s%s%lf", client.lastName,
                 client.firstName, &client.balance );

         /* seek position in file of user-specified record */
         fseek( cfPtr, ( client.acctNum - 1 ) *
                sizeof( struct clientData ), SEEK_SET );

         /* write user-specified information in file */
         fwrite( &client, sizeof( struct clientData ), 1, cfPtr );

         /* enable user to specify another account number */
         printf( "Enter account number\n? " );
         scanf( "%d", &client.acctNum );
```

```
51       } /* end while */
52
53       fclose( cfPtr ); /* fclose closes the file */
54    } /* end else */
55
56    return 0; /* indicates successful termination */
57
58 } /* end main */
```

```
Enter account number ( 1 to 100, 0 to end input )
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
Enter account number
? 88
Enter lastname, firstname, balance
? Smith Dave 258.34
Enter account number
? 33
Enter lastname, firstname, balance
? Dunn Stacey 314.33
Enter account number
? 0
```

Program Output


```
1  /* Fig. 11.15: fig11_15.c
2     Reading a random access file sequentially */
3  #include <stdio.h>
4
5  /* clientData structure definition */
6  struct clientData {
7     int acctNum;          /* account number */
8     char lastName[ 15 ];  /* account last name */
9     char firstName[ 10 ]; /* account first name */
10    double balance;       /* account balance */
11 }; /* end structure clientData */
12
13 int main()
14 {
15    FILE *cfPtr; /* credit.dat file pointer */
16
17    /* create clientData with no information */
18    struct clientData client = { 0, "", "", 0.0 };
19
20    /* fopen opens the file; exits if file cannot be opened */
21    if ( ( cfPtr = fopen( "credit.dat", "rb" ) ) == NULL ) {
22       printf( "File could not be opened.\n" );
23    } /* end if */
```

```
24    else {
25       printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
26               "First Name", "Balance" );
27
28       /* read all records from file (until eof) */
29       while ( !feof( cfPtr ) ) {
30          fread( &client, sizeof( struct clientData ), 1, cfPtr );
31
32          /* display record */
33          if ( client.acctNum != 0 ) {
34             printf( "%-6d%-16s%-11s%10.2f\n",
35                     client.acctNum, client.lastName,
36                     client.firstName, client.balance );
37          } /* end if */
38
39       } /* end while */
40
41       fclose( cfPtr ); /* fclose closes the file */
42    } /* end else */
43
44    return 0; /* indicates successful termination */
45
46 } /* end main */
```



```
Acct  Last Name       First Name    Balance
29    Brown           Nancy          -24.54
33    Dunn            Stacey         314.33
37    Barker          Doug             0.00
88    Smith           Dave           258.34
96    Stone           Sam             34.98
```

Program Output