File position pointer

- File position pointer
  - Indica la ***posizione da cui leggere/scrivere*** i prossimi byte
  - Non un puntatore, ma un ***valore intero*** (specifica la posizione dall'inizio file in termini di numero di byte)
  - È anche chiam
- **`rewind(cfPtr)`**
  - Riposiziona file 0)

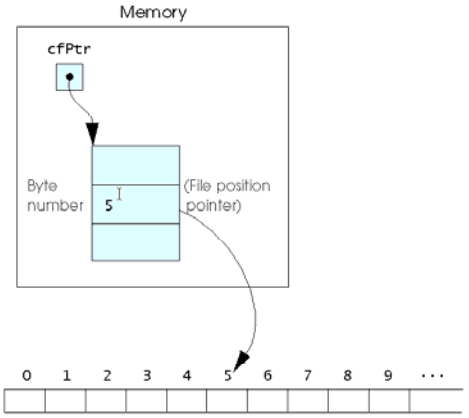


Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

File ad accesso sequenziale

- Se vengono modificati, i dati possono anche essere distrutti (sovrascritti)
- 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:

```
300 Worthington 0.00
```

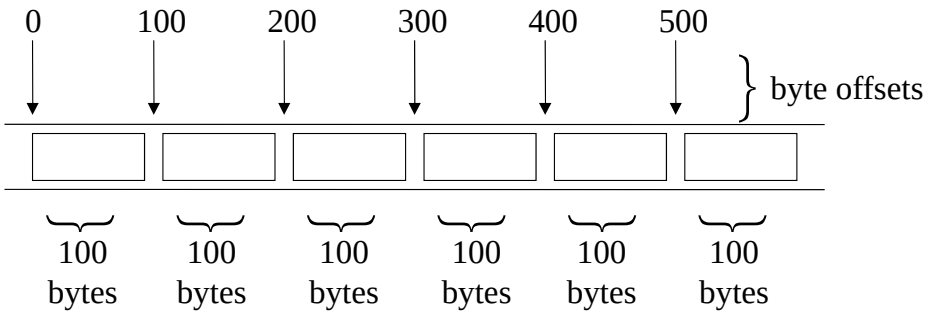
```
300 White 0.00 400 Jones 32.87
```

```
300 Worthington 0.00ones 32.87
```

i dati vengono sovrascritti

File ad accesso diretto

- File ad accesso diretto:
  - Accedono i record individualmente e direttamente
  - Dati possono essere inseriti facilmente senza distruggere altri dati
  - Dati precedenti possono essere modificati o cancellati
- Realizzati usando record di lunghezza fissa
  - In file sequenziali generalmente i record non hanno lunghezza fissa



Posizionamento del puntatore di posizione

- **`fseek()`**

  Setta il file position pointer a una certa posizione

  `fseek(`*`pointer, offset, symbolic_constant`*`);`

  – *pointer* – puntatore al file

  – *offset* – file position pointer (0 è 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

Esempio di uso di file ad accesso diretto

```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 */
```

Esempio di uso di file ad accesso diretto

```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 */
```

Esempio di uso di file ad accesso diretto

```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 {
```

Esempio di uso di file ad accesso diretto

```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
```

Esempio di uso di file ad accesso diretto:

program output