Chapter 11 – File Processing

Outline

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Objectives

- In this chapter, you will learn:
  - To be able to create, read, write and update files.
  - To become familiar with sequential access file processing.
  - To become familiar with random-access file processing.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.1 Introduction

- Data files
  - Can be created, updated, and processed by C programs
  - Are used for permanent storage of large amounts of data
    - Storage of data in variables and arrays is only temporary

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.2 The Data Hierarchy

- Data Hierarchy:
  - Bit – smallest data item
    - Value of `0` or `1`
  - Byte – 8 bits
    - Used to store a character
      - Decimal digits, letters, and special symbols
  - Field – group of characters conveying meaning
    - Example: your name
  - Record – group of related fields
    - Represented by a `struct` or a `class`
    - Example: In a payroll system, a record for a particular employee that contained his/her identification number, name, address, etc.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.2 The Data Hierarchy

- Data Hierarchy (continued):
  - File – group of related records
    - Example: payroll file
  - Database – group of related files

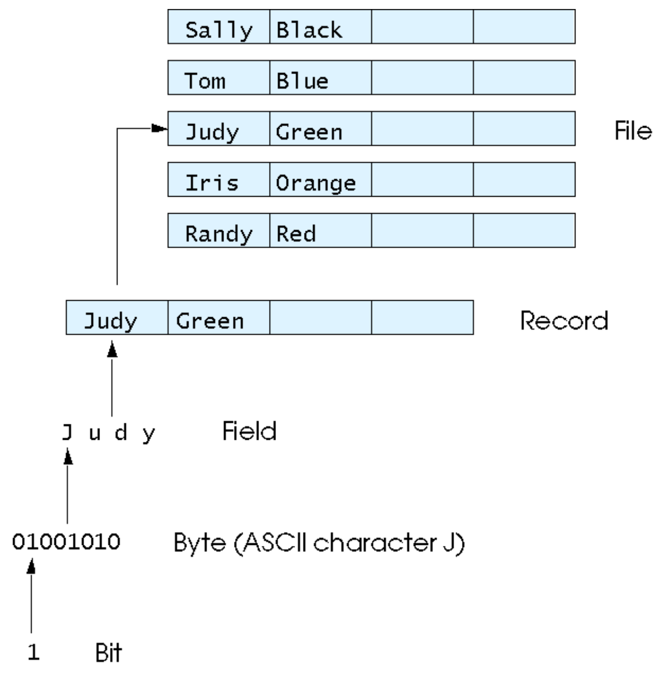


Fig. 11.1 The data hierarchy.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.2 The Data Hierarchy

- Data files
  - Record key
    - Identifies a record to facilitate the retrieval of specific records from a file
  - Sequential file
    - Records typically sorted by key

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.3 Files and Streams

- C views each file as a sequence of bytes
  - File ends with the *end-of-file marker*
    - Or, file ends at a specified byte
- Stream created when a file is opened
  - Provide communication channel between files and programs
  - Opening a file returns a pointer to a `FILE` structure
    - Example file pointers:
    - `stdin` - standard input (keyboard)
    - `stdout` - standard output (screen)
    - `stderr` - standard error (screen)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.3 Files and Streams

- `FILE` structure
  - File descriptor
    - Index into operating system array called the open file table
  - File Control Block (FCB)
    - Found in every array element, system uses it to administer the file

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.3 Files and Streams

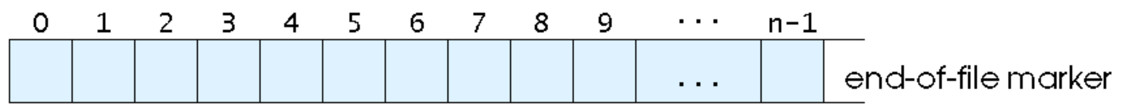


Fig. 11.2 C's view of a file of *n* bytes.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.3 Files and Streams

- Read/Write functions in standard library
  - `fgetc`
    - Reads one character from a file
    - Takes a `FILE` pointer as an argument
    - `fgetc( stdin )` equivalent to `getchar()`
  - `fputc`
    - Writes one character to a file
    - Takes a `FILE` pointer and a character to write as an argument
    - `fputc( 'a', stdout )` equivalent to `putchar( 'a' )`
  - `fgets`
    - Reads a line from a file
  - `fputs`
    - Writes a line to a file
  - `fscanf` / `fprintf`
    - File processing equivalents of `scanf` and `printf`

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```c
/* Fig. 11.3: fig11_03.c
   Create a sequential file */
#include <stdio.h>

int main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

    FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

    /* fopen opens file. Exit program if unable to create file  */
    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
       printf( "File could not be opened\n" );
    } /* end if */
    else {
       printf( "Enter the account, name, and balance.\n" );
       printf( "Enter EOF to end input.\n" );
       printf( "? " );
       scanf( "%d%s%lf", &account, name, &balance );

```

Outline

fig11_03.c (1 of 2)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_03.c (2 of 2)

```
      /* write account, name and balance into file with fprintf */
      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
         printf( "? " );
         scanf( "%d%s%lf", &account, &name, &balance );
      } /* end while */

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

   return 0; /* indicates successful termination */

} /* end main */
```

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
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.4 Creating a Sequential Access File

- C imposes no file structure
  - No notion of records in a file
  - Programmer must provide file structure
- Creating a File
  - `FILE *cfPtr;`
    - Creates a `FILE` pointer called `cfPtr`
  - `cfPtr = fopen(“clients.dat", “w”);`
    - Function `fopen` returns a `FILE` pointer to file specified
    - Takes two arguments – file to open and file open mode
    - If open fails, `NULL` returned

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.4 Creating a Sequential Access File

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.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.4 Creating a Sequential Access File

- `fprintf`
  - Used to print to a file
  - Like `printf`, except first argument is a `FILE` pointer (pointer to the file you want to print in)
- `feof(` *`FILE` pointer* `)`
  - Returns true if end-of-file indicator (no more data to process) is set for the specified file
- `fclose(` *`FILE` pointer* `)`
  - Closes specified file
  - Performed automatically when program ends
  - Good practice to close files explicitly

• Details
- Programs may process no files, one file, or many files
- Each file must have a unique name and should have its own pointer

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.4 Creating a Sequential Access File

Mode	Description
`r`	Open a file for reading.
`w`	Create a file for writing. If the file already exists, discard the current contents.
`a`	Append; open or create a file for writing at end of file.
`r+`	Open a file for update (reading and writing).
`w+`	Create a file for update. If the file already exists, discard the current contents.
`a+`	Append; open or create a file for update; writing is done at the end of the file.
`rb`	Open a file for reading in binary mode.
`wb`	Create a file for writing in binary mode. If the file already exists, discard the current contents.
`ab`	Append; open or create a file for writing at end of file in binary mode.
`rb+`	Open a file for update (reading and writing) in binary mode.
`wb+`	Create a file for update in binary mode. If the file already exists, discard the current contents.
`ab+`	Append; open or create a file for update in binary mode; writing is done at the end of the file.

Fig. 11.6 File open modes.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.5 Reading Data from a Sequential Access File

- Reading a sequential access file
  - Create a `FILE` pointer, link it to the file to read

```
cfPtr = fopen( “clients.dat", "r" );
```

  - Use `fscanf` to read from the file
    - Like `scanf`, except first argument is a `FILE` pointer

```
fscanf( cfPtr, "%d%s%f", &accounnt, name, &balance );
```

  - Data read from beginning to end
  - File position pointer
    - Indicates number of next byte to be read / written
    - Not really a pointer, but an integer value (specifies byte location)
    - Also called byte offset
  - `rewind( cfPtr )`
    - Repositions file position pointer to beginning of file (byte 0)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_07.c (1 of 2)

```
/* Fig. 11.7: fig11_07.c
   Reading and printing a sequential file */
#include <stdio.h>

int main()
{
   int account;     /* account number */
   char name[ 30 ]; /* account name */
   double balance;  /* account balance */

    FILE *cfPtr;     /* cfPtr = clients.dat file pointer */

    /* fopen opens file; exits program if file cannot be opened */
    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
       printf( "File could not be opened\n" );
    } /* end if */
    else { /* read account, name and balance from file */
       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );

       /* while not end of file */
       while ( !feof( cfPtr ) ) {
          printf( "%-10d%-13s%7.2f\n", account, name, balance );
          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
       } /* end while */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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

   return 0; /* indicates successful termination */

} /* 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
```

Outline

fig11_07.c (2 of 2)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_08.c (1 of 5)

```
/* Fig. 11.8: fig11_08.c
   Credit inquiry program */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int request;     /* request number */
   int account;     /* account number */
   double balance;  /* account balance */
   char name[ 30 ]; /* account name */
   FILE *cfPtr;     /* clients.dat file pointer */

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

      /* display request options */
      printf( "Enter request\n"
              " 1 - List accounts with zero balances\n"
              " 2 - List accounts with credit balances\n"
              " 3 - List accounts with debit balances\n"
              " 4 - End of run\n? " );
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_08.c (2 of 5)

```
      scanf( "%d", &request );

      /* process user's request */
      while ( request != 4 ) {

         /* read account, name and balance from file */
         fscanf( cfPtr, "%d%s%lf", &account, name, &balance );

         switch ( request ) {

            case 1:
               printf( "\nAccounts with zero balances:\n" );

               /* read file contents (until eof) */
               while ( !feof( cfPtr ) ) {

                  if ( balance == 0 ) {
                     printf( "%-10d%-13s%7.2f\n",
                             account, name, balance );
                  } /* end if */

                  /* read account, name and balance from file */
                  fscanf( cfPtr, "%d%s%lf",
                          &account, name, &balance );
               } /* end while */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_08.c (3 of 5)

```
            break;

         case 2:
            printf( "\nAccounts with credit balances:\n" );

            /* read file contents (until eof) */
            while ( !feof( cfPtr ) ) {

               if ( balance < 0 ) {
                  printf( "%-10d%-13s%7.2f\n",
                          account, name, balance );
               } /* end if */

               /* read account, name and balance from file */
               fscanf( cfPtr, "%d%s%lf",
                       &account, name, &balance );
            } /* end while */

            break;

         case 3:
            printf( "\nAccounts with debit balances:\n" );

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_08.c (4 of 5)

```
            /* read file contents (until eof) */
            while ( !feof( cfPtr ) ) {

               if ( balance > 0 ) {
                  printf( "%-10d%-13s%7.2f\n",
                          account, name, balance );
               } /* end if */

               /* read account, name and balance from file */
               fscanf( cfPtr, "%d%s%lf",
                       &account, name, &balance );
            } /* end while */

            break;

      } /* end switch */

      rewind( cfPtr ); /* return cfPtr to beginning of file */

      printf( "\n? " );
      scanf( "%d", &request );
   } /* end while */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_08.c (5 of 5)

```
      printf( "End of run.\n" );
      fclose( cfPtr ); /* fclose closes the file */
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */
```

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.
```

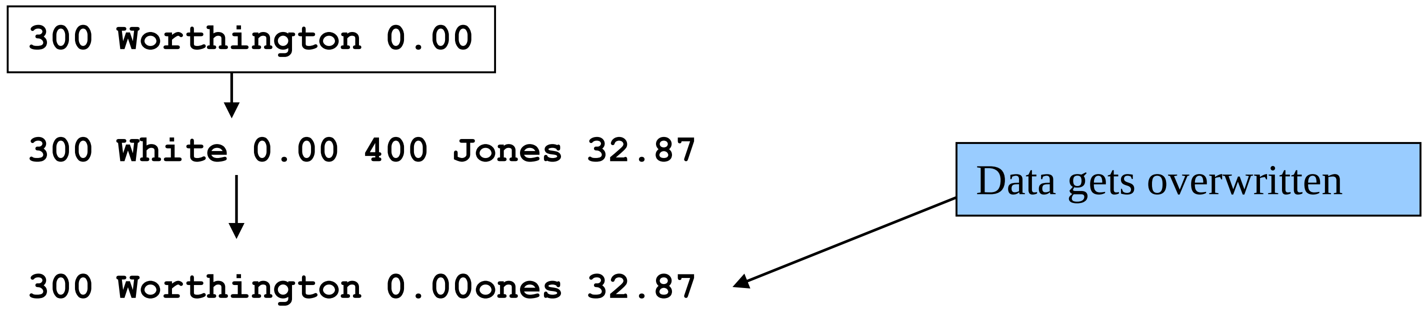
© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.5 Reading Data from a Sequential Access File

- Sequential access file
  - Cannot be modified without the risk of destroying other data
  - Fields can vary in size
    - Different representation in files and screen than internal representation
    - `1`, `34`, `-890` are all `int`s, but have different sizes on disk

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

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

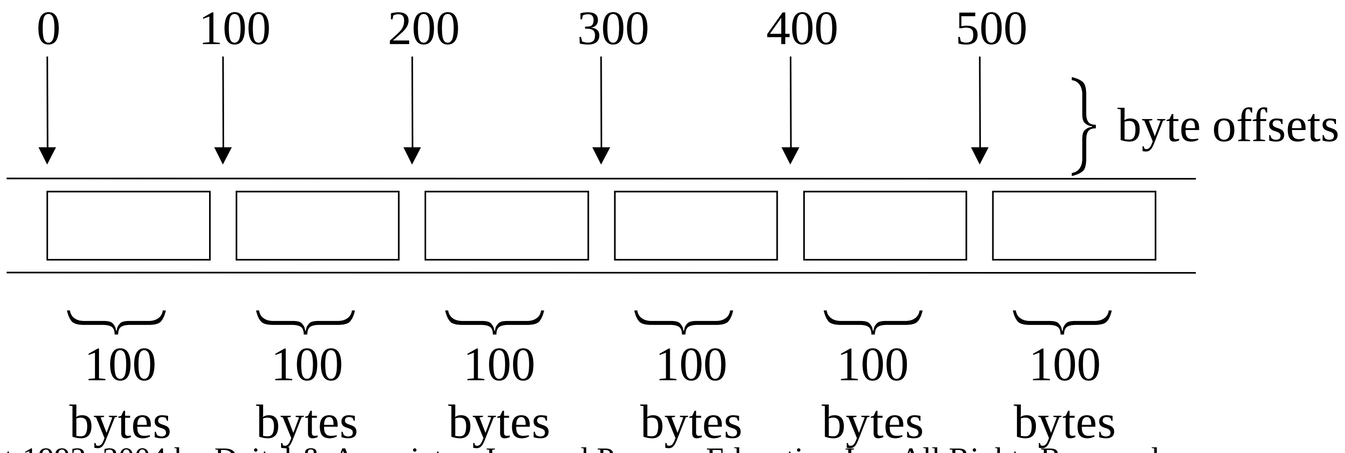


```
300 Worthington 0.00

300 White 0.00 400 Jones 32.87

300 Worthington 0.00ones 32.87
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.6 Random-Access Files

- Random access files
  - Access individual records without searching through other records
  - Instant access to records in a file
  - Data can be inserted without destroying other data
  - Data previously stored can be updated or deleted without overwriting
- Implemented using fixed length records
  - Sequential files do not have fixed length records



© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.7 Creating a Randomly Accessed File

- Data in random access files
  - Unformatted (stored as "raw bytes")
    - All data of the same type (**`int`**s, for example) uses the same amount of memory
    - All records of the same type have a fixed length
    - Data not human readable

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.7 Creating a Randomly Accessed File

- Unformatted I/O functions
  - `fwrite`
    - Transfer bytes from a location in memory to a file
  - `fread`
    - Transfer bytes from a file to a location in memory
  - Example:

    ```
    fwrite( &number, sizeof( int ), 1, myPtr );
    ```

    - `&number` – Location to transfer bytes from
    - `sizeof( int )` – Number of bytes to transfer
    - `1` – For arrays, number of elements to transfer
      - In this case, "one element" of an array is being transferred
    - `myPtr` – File to transfer to or from

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.7 Creating a Randomly Accessed File

- Writing `structs`

```
fwrite( &myObject, sizeof (struct myStruct), 1,
  myPtr );
```

  - `sizeof` – returns size in bytes of object in parentheses
- To write several array elements
  - Pointer to array as first argument
  - Number of elements to write as third argument

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_11.c (1 of 2)

```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 */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_11.c (2 of 2)

```
   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 */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.8 Writing Data Randomly to a Randomly Accessed File

- `fseek`
  - Sets file position pointer to a specific position
  - `fseek(` *pointer*, *offset*, *symbolic_constant* `);`
    - *pointer* – pointer to file
    - *offset* – file position pointer (0 is first location)
    - *symbolic_constant* – specifies where in file we are reading from
    - `SEEK_SET` – seek starts at beginning of file
    - `SEEK_CUR` – seek starts at current location in file
    - `SEEK_END` – seek starts at end of file

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_12.c (1 of 3)

```
/* 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 {

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_12.c (2 of 3)

```
   /* 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 );
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```c
      } /* end while */

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

   return 0; /* indicates successful termination */

} /* end main */
```

fig11_12.c (3 of 3)

Program Output

```
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
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.8 Writing Data Randomly to a Randomly Accessed File

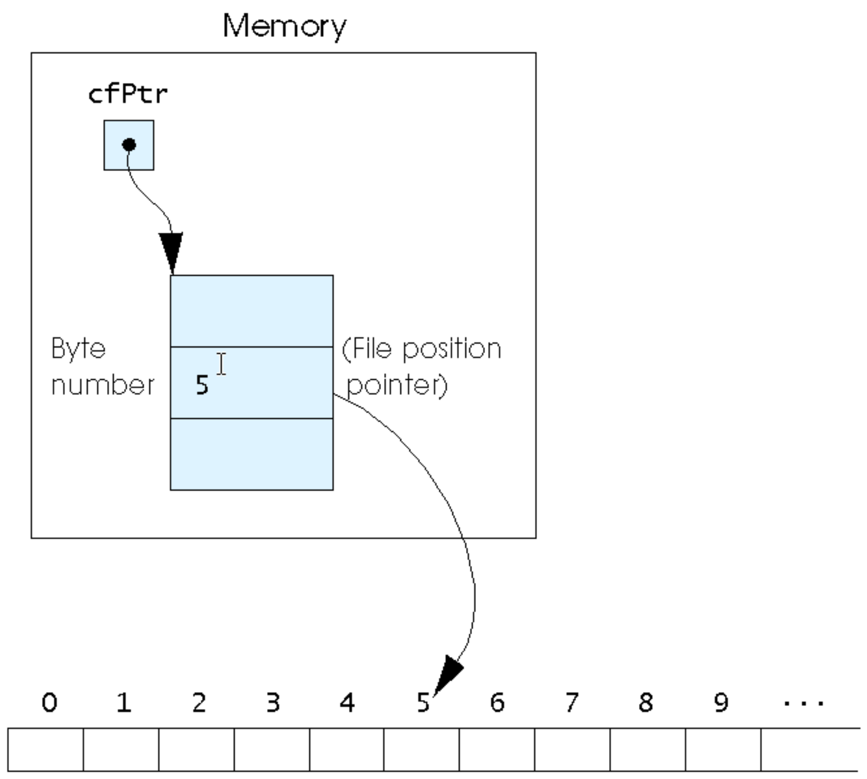


Fig. 11.14 The file position pointer indicating an offset of 5 bytes from the beginning of the file.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.9 Reading Data Randomly from a Randomly Accessed File

- `fread`
  - Reads a specified number of bytes from a file into memory

    ```
    fread( &client, sizeof (struct clientData), 1,
      myPtr );
    ```
  - Can read several fixed-size array elements
    - Provide pointer to array
    - Indicate number of elements to read
  - To read multiple elements, specify in third argument

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_15.c (1 of 2)

```c
/* Fig. 11.15: fig11_15.c
   Reading a random access 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()
{
   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 */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_15.c (2 of 2)

```
   else {
      printf( "%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
              "First Name", "Balance" );

      /* read all records from file (until eof) */
      while ( !feof( cfPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1, cfPtr );

         /* display record */
         if ( client.acctNum != 0 ) {
            printf( "%-6d%-16s%-11s%10.2f\n",
                    client.acctNum, client.lastName,
                    client.firstName, client.balance );
         } /* end if */

      } /* end while */

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

   return 0; /* indicates successful termination */

} /* end main */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Program Output

```
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
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

11.10 Case Study: A Transaction Processing Program

- This program
  - Demonstrates using random access files to achieve instant access processing of a bank’s account information
- We will
  - Update existing accounts
  - Add new accounts
  - Delete accounts
  - Store a formatted listing of all accounts in a text file

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (1 of 11)

```c
/* Fig. 11.16: fig11_16.c
   This program reads a random access file sequentially, updates data
   already written to the file, creates new data to be placed in the
   file, and deletes data previously in the 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 */

 /* prototypes */
 int enterChoice( void );
 void textFile( FILE *readPtr );
 void updateRecord( FILE *fPtr );
 void newRecord( FILE *fPtr );
 void deleteRecord( FILE *fPtr );

 int main()
 {
    FILE *cfPtr; /* credit.dat file pointer */
    int choice;  /* user's choice */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_16.c (2 of 11)

```
   /* 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 {

      /* enable user to specify action */
      while ( ( choice = enterChoice() ) != 5 ) {

         switch ( choice ) {

            /* create text file from record file */
            case 1:
               textFile( cfPtr );
               break;

            /* update record */
            case 2:
               updateRecord( cfPtr );
               break;

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (3 of 11)

```
            /* create record */
            case 3:
               newRecord( cfPtr );
               break;

            /* delete existing record */
            case 4:
               deleteRecord( cfPtr );
               break;

            /* display message if user does not select valid choice */
            default:
               printf( "Incorrect choice\n" );
               break;

         } /* end switch */

      } /* end while */

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

   return 0; /* indicates successful termination */

} /* end main */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (4 of 11)

```
/* create formatted text file for printing */
void textFile( FILE *readPtr )
{
   FILE *writePtr; /* accounts.txt file pointer */

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

   /* fopen opens the file; exits if file cannot be opened */
   if ( ( writePtr = fopen( "accounts.txt", "w" ) ) == NULL ) {
      printf( "File could not be opened.\n" );
   } /* end if */
   else {
      rewind( readPtr ); /* sets pointer to beginning of record file */
      fprintf( writePtr, "%-6s%-16s%-11s%10s\n",
               "Acct", "Last Name", "First Name","Balance" );

      /* copy all records from record file into text file */
      while ( !feof( readPtr ) ) {
         fread( &client, sizeof( struct clientData ), 1, readPtr );

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_16.c (5 of 11)

```
         /* write single record to text file */
         if ( client.acctNum != 0 ) {
            fprintf( writePtr, "%-6d%-16s%-11s%10.2f\n",
                     client.acctNum, client.lastName,
                     client.firstName, client.balance );
          } /* end if */

       } /* end while */

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

} /* end function textFile */

/* update balance in record */
void updateRecord( FILE *fPtr )
{
   int account;        /* account number */
   double transaction; /* account transaction */

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

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (6 of 11)

```
   /* obtain number of account to update */
   printf( "Enter account to update ( 1 - 100 ): " );
   scanf( "%d", &account );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   /* display error if account does not exist */
   if ( client.acctNum == 0 ) {
      printf( "Acount #%d has no information.\n", account );
   } /* end if */
   else { /* update record */
      printf( "%-6d%-16s%-11s%10.2f\n\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );

      /* request user to specify transaction */
      printf( "Enter charge ( + ) or payment ( - ): " );
      scanf( "%lf", &transaction );
      client.balance += transaction; /* update record balance */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (7 of 11)

```
      printf( "%-6d%-16s%-11s%10.2f\n",
              client.acctNum, client.lastName,
              client.firstName, client.balance );

      /* move file pointer to correct record in file */
      fseek( fPtr, ( account - 1 ) * sizeof( struct clientData ),
             SEEK_SET );

      /* write updated record over old record in file */
      fwrite( &client, sizeof( struct clientData ), 1, fPtr );
   } /* end else */

} /* end function updateRecord */

/* delete an existing record */
void deleteRecord( FILE *fPtr )
{
   /* create two clientDatas and initialize blankClient */
   struct clientData client;
   struct clientData blankClient = { 0, "", "", 0 };

   int accountNum; /* account number */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (8 of 11)

```
   /* obtain number of account to delete */
   printf( "Enter account number to delete ( 1 - 100 ): " );
   scanf( "%d", &accountNum );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

   /* display error if record does not exist */
   if ( client.acctNum == 0 ) {
      printf( "Account %d does not exist.\n", accountNum );
   } /* end if */
   else { /* delete record */

      /* move file pointer to correct record in file */
      fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
         SEEK_SET );

      /* replace existing record with blank record */
      fwrite( &blankClient,
              sizeof( struct clientData ), 1, fPtr );
   } /* end else */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

fig11_16.c (9 of 11)

```
} /* end function deleteRecord */

/* create and insert record */
void newRecord( FILE *fPtr )
{
   /* create clientData with no information */
   struct clientData client = { 0, "", "", 0.0 };

   int accountNum; /* account number */

   /* obtain number of account to create */
   printf( "Enter new account number ( 1 - 100 ): " );
   scanf( "%d", &accountNum );

   /* move file pointer to correct record in file */
   fseek( fPtr, ( accountNum - 1 ) * sizeof( struct clientData ),
          SEEK_SET );

   /* read record from file */
   fread( &client, sizeof( struct clientData ), 1, fPtr );

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (10 of 11)

```
   /* display error if account previously exists */
   if ( client.acctNum != 0 ) {
      printf( "Account #%d already contains information.\n",
              client.acctNum );
   } /* end if */
   else { /* create record */

      /* user enters last name, first name and balance */
      printf( "Enter lastname, firstname, balance\n? " );
      scanf( "%s%s%lf", &client.lastName, &client.firstName,
             &client.balance );

      client.acctNum = accountNum;

      /* move file pointer to correct record in file */
      fseek( fPtr, ( client.acctNum - 1 ) *
             sizeof( struct clientData ), SEEK_SET );

      /* insert record in file */
      fwrite( &client,
              sizeof( struct clientData ), 1, fPtr );
   } /* end else */

} /* end function newRecord */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Outline

fig11_16.c (11 of 11)

```
/* enable user to input menu choice */
int enterChoice( void )
{
   int menuChoice; /* variable to store user's choice */

   /* display available options */
   printf( "\nEnter your choice\n"
           "1 - store a formatted text file of acounts called\n"
           "    \"accounts.txt\" for printing\n"
           "2 - update an account\n"
           "3 - add a new account\n"
           "4 - delete an account\n"
           "5 - end program\n? " );

   scanf( "%d", &menuChoice ); /* receive choice from user */

   return menuChoice;

} /* end function enterChoice */
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Program Output

```
After choosing option 1 accounts.txt contains:

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
```

```
After choosing option 2 accounts.txt contains:

Enter account to update ( 1 - 100 ): 37
37    Barker           Doug             0.00

Enter charge ( + ) or payment ( - ): +87.99
37    Barker           Doug            87.99
```

```
After choosing option 3 accounts.txt contains:

Enter new account number ( 1 - 100 ): 22
Enter lastname, firstname, balance
? Johnston Sarah 247.45
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.