11

C File Processing

OBJECTIVES

In this chapter you will learn:

- To create, read, write and update files.
- Sequential access file processing.

Outline

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**

11.2 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.**

11.2 Data Hierarchy

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

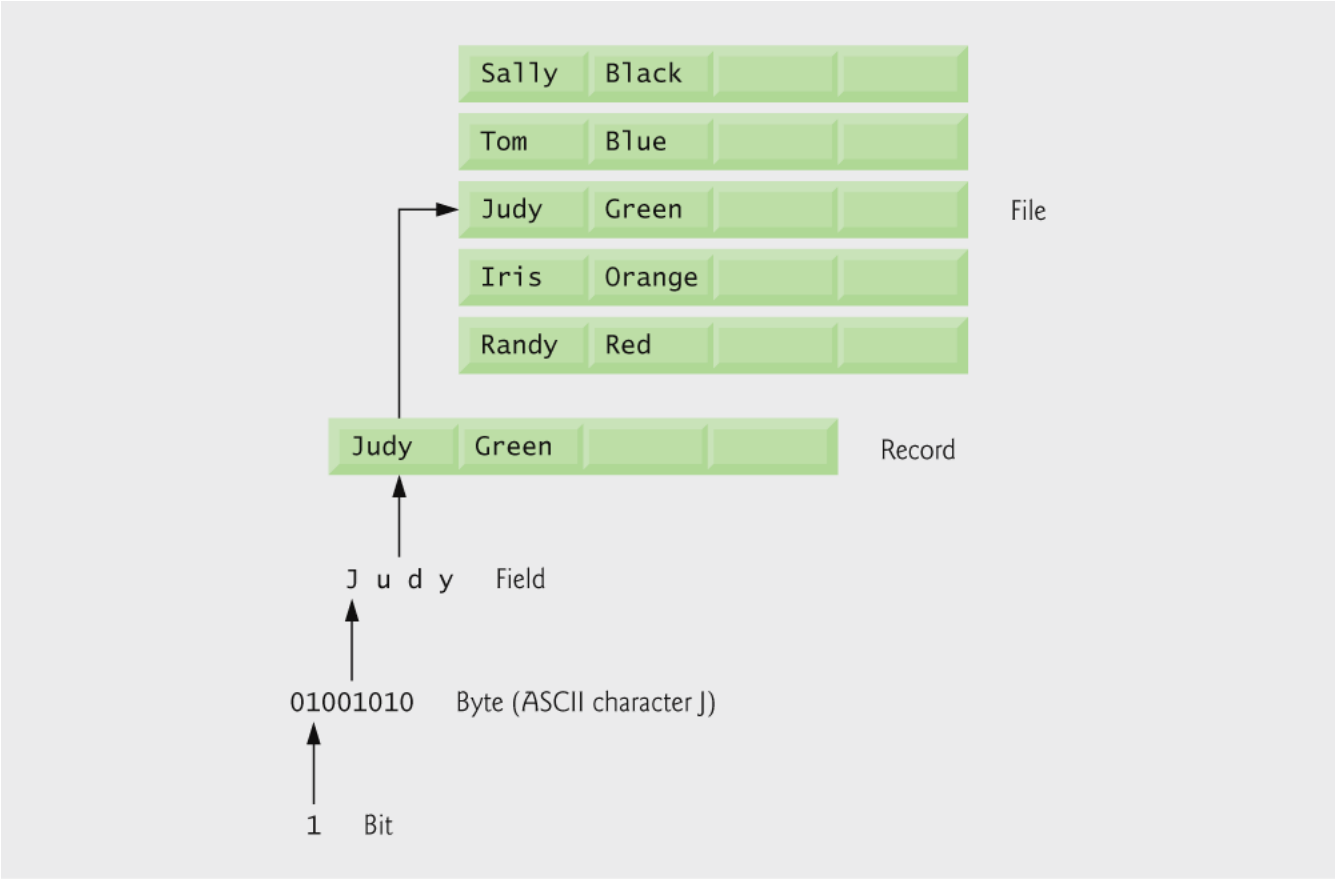

Fig. 11.1 | Data hierarchy.

11.2 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**

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)

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**

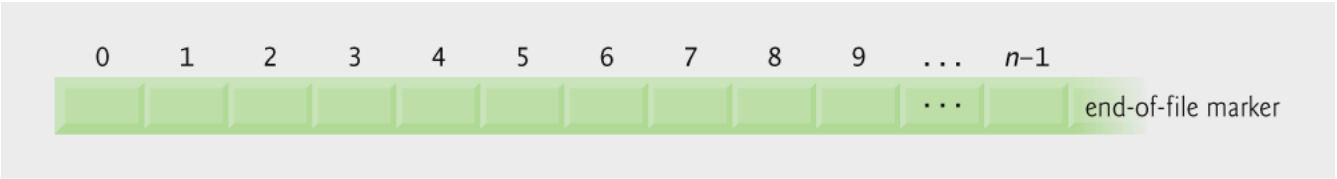

Fig. 11.2 | C's view of a file of *n* bytes.

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`

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

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

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( void )
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
```

FILE pointer definition creates new file pointer

fopen function opens a file; **w** argument means the file is opened for writing

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 */
```

feof returns true when end of file is reached

fprintf writes a string to a file

fclose closes a file

fig11_03.c

(2 of 2)

```
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
```

Common Programming Error 11.1

Opening an existing file for writing (`"w"`) when, in fact, the user wants to preserve the file, discards the contents of the file without warning.

Common Programming Error 11.2

Forgetting to open a file before attempting to reference it in a program is a logic error.

Common Programming Error 11.3

Using the wrong file pointer to refer to a file is a logic error.

Error-Prevention Tip 11.1

Be sure that calls to file processing functions in a program contain the correct file pointers.

Operating system	Key combination
Linux/Mac OS X/UNIX	*<Ctrl> d*
Windows	*<Ctrl> z*

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

Good Programming Practice 11.1

Explicitly close each file as soon as it is known that the program will not reference the file again.

Performance Tip 11.1

Closing a file can free resources for which other users or programs may be waiting.

Common Programming Error 11.4

Opening a nonexistent file for reading is an error.

Common Programming Error 11.5

Opening a file for reading or writing without having been granted the appropriate access rights to the file (this is operating-system dependent) is an error.

Common Programming Error 11.6

Opening a file for writing when no disk space is available is an error.

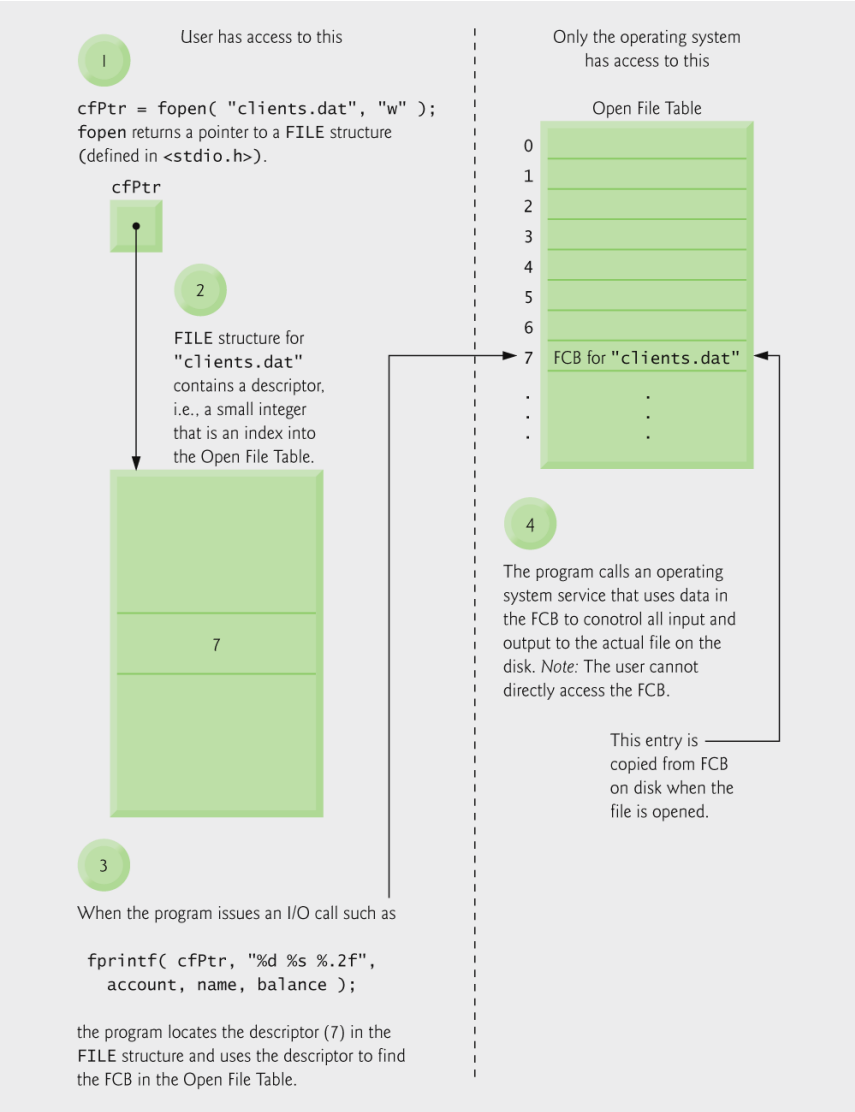

Fig. 11.5 | Relationship between **`FILE`** pointers, **`FILE`** structures and FCBs.

Common Programming Error 11.7

Opening a file with the incorrect file mode is a logic error. For example, opening a file in write mode (`"w"`) when it should be opened in update mode (`"r+"`) causes the contents of the file to be discarded.

Mode	Description
`r`	Open an existing 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 the end of the file.
`r+`	Open an existing 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 an existing 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 the end of the file in binary mode.
`rb+`	Open an existing 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 opening modes.

Error-Prevention Tip 11.2

Open a file only for reading (and not update) if the contents of the file should not be modified. This prevents unintentional modification of the file's contents. This is another example of the principle of least privilege.

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)

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( void )
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
```

fopen function opens a file; **r** argument means the file is opened for reading

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

`fscanf` function reads a string from a file

```
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)

Outline

fig11_08.c

(1 of 4)

```
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( void )
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
```

Outline

fig11_08.c

(2 of 4)

```
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
52                break;
53
```

Outline

fig11_08.c

(3 of 4)

```
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
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
```

Outline

fig11_08.c

(4 of 4)

```
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
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 */
```

rewind function moves the file pointer back to the beginning of the file

Outline

```
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.
```

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**