CC-112 Programming Fundamentals

File Processing

Nazar Khan
Department of Computer Science
University of the Punjab

Introduction

- Storage of data in variables and arrays is temporary – such data is lost when a program terminates.
- Files are used for long-term retention of data.
- Computers store files on *secondary storage* devices, such as hard drives, solid-state drives, flash drives and DVDs.
- In this lecture, we study how data files are *created, updated and processed* by C programs.
- We consider both *sequential-access* and *random-access* file processing.

Files and Streams

- C views each file simply as a *sequential stream of bytes*.
- Each file ends either with an *end-of-file* marker or at a specific byte number recorded in a system-maintained, administrative data structure – this is determined by each platform and is hidden from you.

0	1	2	3	4	5	. . .	n-1	
								end-of-file marker

Files and Streams

- When a file is opened, a *stream* is associated with it.
- A stream is a *communication channel a file and a program*.
- Three streams are automatically opened when program execution begins.
  1. the *standard input* (which receives input from the keyboard),
  2. the *standard output* (which displays output on the screen) and
  3. the *standard error* (which displays error messages on the screen).

The `FILE` structure

- Opening a file returns a pointer to a `FILE structure` (defined in `<stdio.h>`)
- It contains information used to process the file.
  - file descriptor – an integer index into an operating-system array called the *open file table*.
  - Each array element contains a *file control block (FCB)* – information that the operating system uses to administer a particular file.
- The standard input, standard output and standard error are manipulated using
  - `FILE * stdin`,
  - `FILE * stdout`, and
  - `FILE * stderr`,

  respectively.

Sequential-access files

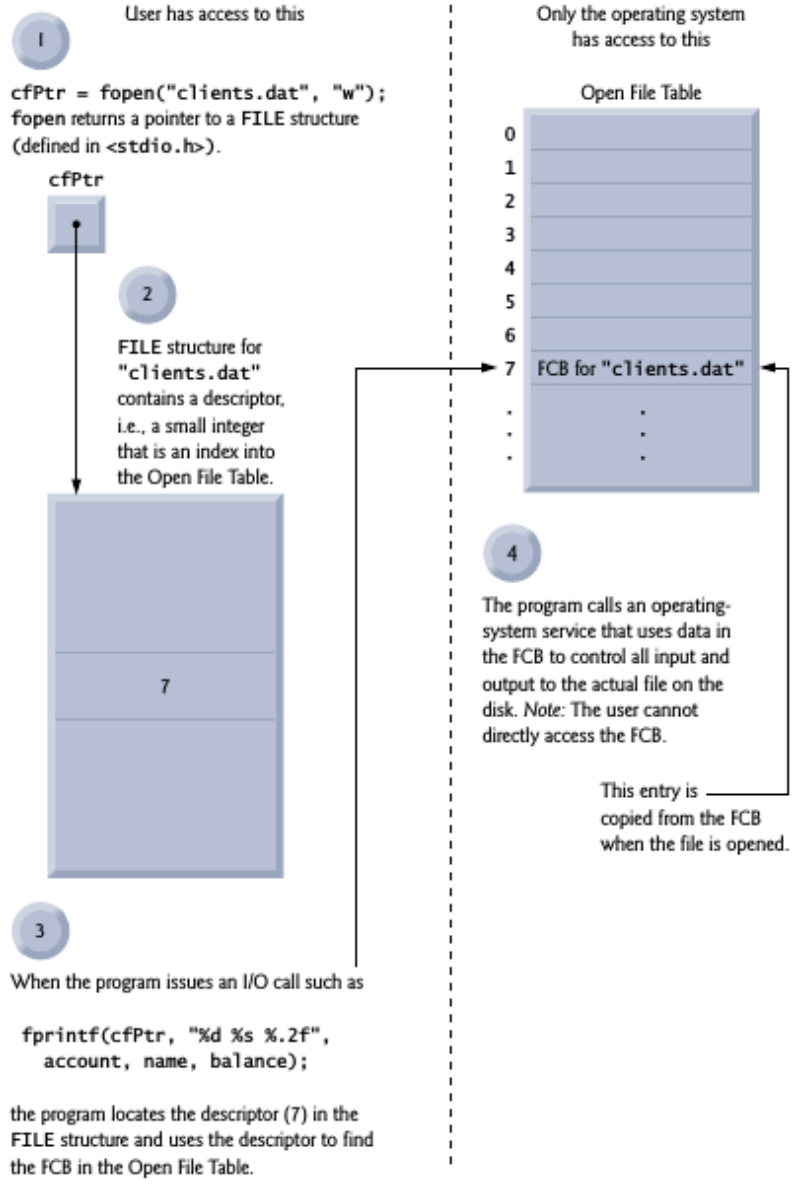
```c
// Creating a sequential file
#include <stdio.h>

int main(void)
{
   FILE *cfPtr; // cfPtr = clients.txt file pointer

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

      unsigned int account; // account number
      char name[30]; // account name
      double balance; // account balance

      scanf("%d%29s%lf", &account, name, &balance);

      // write account, name and balance into file with fprintf
      while (!feof(stdin)) {
         fprintf(cfPtr, "%d %s %.2f\n", account, name, balance);
         printf("%s", "? ");
         scanf("%d%29s%lf", &account, name, &balance);
      }
```

Sequential-access files

```
      fclose(cfPtr); // fclose closes file
   }
}
```

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
```



Relationship between FILE pointers, FILE structures and FCBs.

File opening modes

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`	Open or create a file for writing at the end of the file—i.e., write operations *append* data to the file.
`r+`	Open an existing file for update (reading and writing).
`w+`	Create a file for reading and writing. If the file already exists, *discard* the current contents.
`a+`	Open or create a file for reading and updating; all writing is done at the end of the file—i.e., write operations *append* data to 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.

Reading from a sequential-access file

```c
// Reading and printing a sequential file
#include <stdio.h>

int main(void)
{
   FILE *cfPtr; // cfPtr = clients.txt file pointer

   // fopen opens file; exits program if file cannot be opened
   if ((cfPtr = fopen("clients.txt", "r")) == NULL) {
      puts("File could not be opened");
   }
   else { // read account, name and balance from file
      unsigned int account; // account number
      char name[30]; // account name
      double balance; // account balance

      printf("%-10s%-13s%s\n", "Account", "Name", "Balance");
      fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);

      // while not end of file
      while (!feof(cfPtr)) {
         printf("%-10d%-13s%7.2f\n", account, name, balance);
         fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);
      }

      fclose(cfPtr); // fclose closes the file
   }
}
```

Reading from a sequential-access file

Output

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

Reading from a sequential-access file multiple times

Via rewind function

```
// Credit inquiry program
#include <stdio.h>

// function main begins program execution
int main(void)
{
   FILE *cfPtr; // clients.txt file pointer

   // fopen opens the file; exits program if file cannot be opened
   if ((cfPtr = fopen("clients.txt", "r")) == NULL) {
      puts("File could not be opened");
   }
   else {
      // display request options
      printf("%s", "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? ");
      unsigned int request; // request number
      scanf("%u", &request);

      // process user's request
      while (request != 4) {
         unsigned int account; // account number
         double balance; // account balance
         char name[30]; // account name
```

Reading from a sequential-access file multiple times

Via `rewind` function

```
// read account, name and balance from file
fscanf(cfPtr, "%d%29s%lf", &account, name, &balance);

switch (request) {
   case 1:
      puts("\nAccounts with zero balances:");

      // read file contents (until eof)
      while (!feof(cfPtr)) {
         // output only if balance is 0
         if (balance == 0) {
            printf("%-10d%-13s%7.2f\n",
               account, name, balance);
         }

         // read account, name and balance from file
         fscanf(cfPtr, "%d%29s%lf",
            &account, name, &balance);
      }

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

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

Reading from a sequential-access file multiple times

Via `rewind` function

```
            // output only if balance is less than 0
            if (balance < 0) {
               printf("%-10d%-13s%7.2f\n",
                  account, name, balance);
            }

            // read account, name and balance from file
            fscanf(cfPtr, "%d%29s%lf",
               &account, name, &balance);
         }

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

         // read file contents (until eof)
         while (!feof(cfPtr)) {
            // output only if balance is greater than 0
            if (balance > 0) {
               printf("%-10d%-13s%7.2f\n",
                  account, name, balance);
            }

            // read account, name and balance from file
            fscanf(cfPtr, "%d%29s%lf",
               &account, name, &balance);
         }
```

Reading from a sequential-access file multiple times

Via rewind function

```
                break;
            }

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

            printf("%s", "\n? ");
            scanf("%d", &request);
        }

        puts("End of run.");
        fclose(cfPtr); // fclose closes the file
    }
}
```

Reading from a sequential-access file multiple times

Via `rewind` function

Output

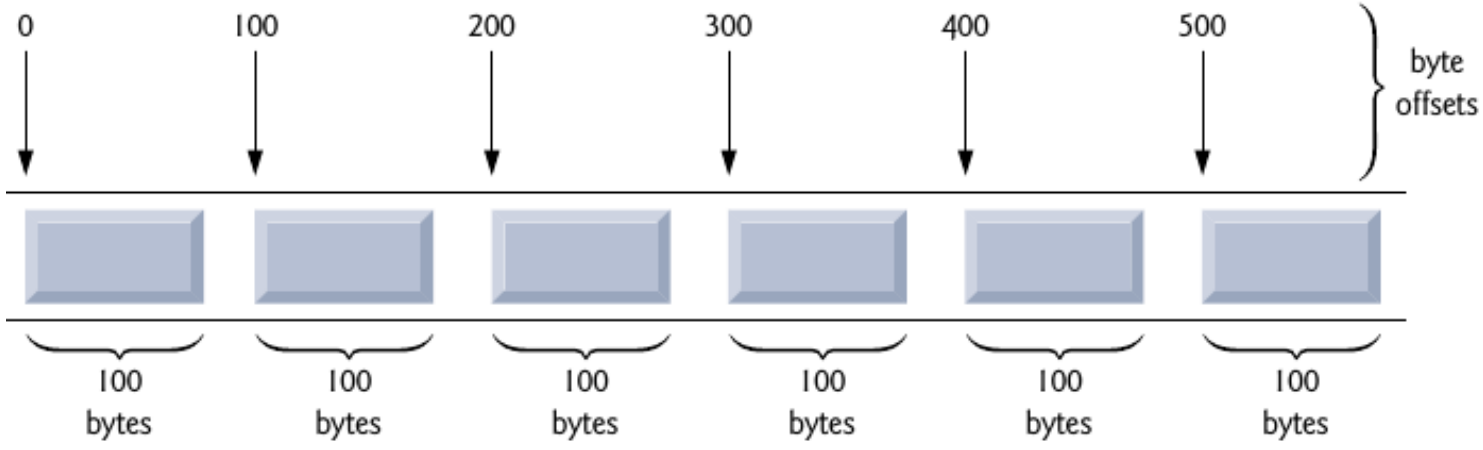
```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

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.
```

Updating sequential-access files can be problematic

- Changing `300 White 0.00` to `300 Shakespeare 0.00` will start *overwriting the next record in the file* by `speare 0.00`.
- This is because we are saving everything as variable-sized *text*.
- Solution 1: i) copy every thing before `300 White 0.00`, ii) write `300 Shakespeare 0.00`, and iii) copy all the rest.
  - Problem: processed every record in the file to update one record.

Random-access files

- Solution 2: save every record as fixed-size binary numbers.
- When every record has the same length, say 100 bytes, the n-th record can be *directly accessed* by an offset of $100(n-1)$.
- This is called *random-access* of records.



- Fixed-length records enable data to be inserted in a random-access file without destroying other data in the file.
- Data stored previously can also be updated or deleted without rewriting the entire file.

Creating a random-access file

```
// Creating a random-access file sequentially
#include <stdio.h>

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

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

   // fopen opens the file; exits if file cannot be opened
   if ((cfPtr = fopen("accounts.dat", "wb")) == NULL) {
      puts("File could not be opened.");
   }
   else {
      // create clientData with default information
      struct clientData blankClient = {0, "", "", 0.0};

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

Creating a random-access file

```
      fclose (cfPtr); // fclose closes the file
   }
}
```

Writing data *randomly* to a random-access file

fseek and *fwrite*

```
// Writing data randomly to a random-access file
#include <stdio.h>

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

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

   // fopen opens the file; exits if file cannot be opened
   if ((cfPtr = fopen("accounts.dat", "rb+")) == NULL) {
      puts("File could not be opened.");
   }
   else {
      // create clientData with default information
      struct clientData client = {0, "", "", 0.0};

      // require user to specify account number
      printf("%s", "Enter account number"
         " (1 to 100, 0 to end input): ");
      scanf("%d", &client.acctNum);
```

Writing data *randomly* to a random-access file

fseek and *fwrite*

```
      // user enters information, which is copied into file
      while (client.acctNum != 0) {
         // user enters last name, first name and balance
         printf("%s", "Enter lastname, firstname, balance: ");

         // set record lastName, firstName and balance value
         fscanf(stdin, "%14s%9s%lf", client.lastName,
            client.firstName, &client.balance);

         // seek position in file to 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 input another account number
         printf("%s", "Enter account number: ");
         scanf("%d", &client.acctNum);
      }

      fclose(cfPtr); // fclose closes the file
   }
}
```

Writing data *randomly* to a random-access file

fseek and *fwrite*

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
```

Reading from a random-access file

fread

```
// Reading a random-access file sequentially
#include <stdio.h>

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

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

   // fopen opens the file; exits if file cannot be opened
   if ((cfPtr = fopen("accounts.dat", "rb")) == NULL) {
      puts("File could not be opened.");
   }
   else {
      printf("%-6s%-16s%-11s%10s\n", "Acct", "Last Name",
         "First Name", "Balance");

      // read all records from file (until eof)
      while (!feof(cfPtr)) {
         // create clientData with default information
         struct clientData client = {0, "", "", 0.0};
```

Reading from a random-access file

fread

```
         int result = fread(&client, sizeof(struct clientData), 1, cfPtr);

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

      fclose(cfPtr); // fclose closes the file
   }
}
```

Reading from a random-access file

fread

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
```