Programming Fundamentals (CT-175) Lab 13

File Handling in C Language.

Objectives

The objective of this lab is to enable students handle files in C language.

Tools Required

DevC++

Course Coordinator –
Course Instructor –
Lab Instructor –
Prepared By Department of Computer Science and Information Technology
NED University of Engineering and Technology

Filing

A file is a container in computer storage devices used for storing data. In this lab, we will be learning about file handling in C. We will learn to handle standard I/O in C using fprintf(), fscanf(), fread(), fwrite(), fseek() etc. with the help of examples.

Why files are needed?

When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates. If you have to enter a large number of data, it will take a lot of time to enter them all. However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C. You can easily move your data from one computer to another without any changes.

Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

1. Text files

Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad. When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents. They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

2. Binary files

Binary files are mostly the .bin files in your computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's). They can hold a higher amount of data, are not readable easily, and provides better security than text files.

File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

```
FILE *fptr;
```

Opening a file - for creation and edit

Opening a file is performed using the fopen() function defined in the stdio.h header file. The syntax for opening a file in standard I/O is:

```
ptr = fopen("fileopen","mode");
```

For example,

```
fopen("E:\\cprogram\\newprogram.txt","w");

fopen("E:\\cprogram\\oldprogram.bin","rb");
```

Let's suppose the file newprogram.txt doesn't exist in the location E:\cprogram. The first function creates a new file named newprogram.txt and opens it for writing as per the mode 'w'. The writing mode allows you to create and edit (overwrite) the contents of the file. Now let's suppose the second binary file oldprogram.bin exists in the location E:\cprogram. The second function opens the existing file for reading in binary mode 'rb'. The reading mode only allows you to read the file, you cannot write into the file.

Opening Modes in Standard I/O

Mode	Meaning of Mode	During Inexistence of file
R	Open for reading.	If the file does not exist, fopen() returns NULL.
Rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
W	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
Wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
A	Open for append. Data is added to the end of the file.	If the file does not exist, it will be created.
Ab	Open for append in binary mode. Data is added to the end of the file.	If the file does not exist, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both	If the file exists, its contents are overwritten.

Opening Modes in Standard I/O

Mode	Meaning of Mode	During Inexistence of file
	reading and writing.	If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exist, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

Closing a File

The file (both text and binary) should be closed after reading/writing. Closing a file is performed using the fclose() function.

```
fclose(fptr);
```

Here, fptr is a file pointer associated with the file to be closed.

Reading and writing to a text file

For reading and writing to a text file, we use the functions fprintf() and fscanf(). They are just the file versions of printf() and scanf(). The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.

Example 1: Write to a text file

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num;
   FILE *fptr;

   // use appropriate location if you are using MacOS or Linux
   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }

   printf("Enter num: ");
   scanf("%d",&num);

   fprintf(fptr,"%d",num);
   fclose(fptr);

   return 0;
}
```

This program takes a number from the user and stores in the file program.txt. After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

Example 2: Read from a text file

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.txt","r")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   fscanf(fptr,"%d", &num);

   printf("Value of n=%d", num);
   fclose(fptr);

   return 0;
}
```

This program reads the integer present in the program.txt file and prints it onto the screen. If you successfully created the file from Example 1, running this program will get you the integer you entered. Other functions like fgetchar(), fputc() etc. can be used in a similar way.

Reading and writing to a binary file

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.

Writing to a binary file

To write into a binary file, you need to use the fwrite() function. The functions take four arguments:

1. address of data to be written in the disk
2. size of data to be written in the disk
3. number of such type of data
4. pointer to the file where you want to write.

```
fwrite(addressData, sizeData, numbersData, pointerToFile);
```

Example 3: Write to a binary file using fwrite()

```
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.bin","wb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      num.n1 = n;
      num.n2 = 5*n;
      num.n3 = 5*n + 1;
      fwrite(&num, sizeof(struct threeNum), 1, fptr);
   }
   fclose(fptr);

   return 0;
}
```

In this program, we create a new file program.bin in the C drive. We declare a structure threeNum with three numbers - n1, n2 and n3, and define it in the main function as num. Now, inside the for loop, we store the value into the file using fwrite(). The first parameter takes the address of num and the second parameter takes the size of the structure threeNum. Since we're only inserting one instance of num, the third parameter is 1. And, the last parameter *fptr points to the file we're storing the data. Finally, we close the file.

Reading from a binary file

Function fread() also take 4 arguments similar to the fwrite() function as above.

```
fread(addressData, sizeData, numbersData, pointerToFile);
```

Example 4: Read from a binary file using fread()

```
#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2, num.n3);
   }
   fclose(fptr);

   return 0;
}
```

In this program, you read the same file program.bin and loop through the records one by one. In simple terms, you read one threeNum record of threeNum size from the file pointed by *fptr into the structure num.

EXERCISE

1. Write a program to take input from user and save that text in a file as encrypted text (using Shift cipher along with K=3). Then retrieve the encrypted text from the same file and display the original text.
2. Write a C program to keep records and perform statistical analysis for a class of 20 students. The information of each student contains ID, Name, Sex, quizzes Scores (2 quizzes per semester), mid-term score, final score, and total score. All the records must be store in the file and you must read the scores <50, <80 and <100 until users selects the end file option.
3. You're the owner of a hardware store and need to keep an inventory that can tell you what tools you have, how many you have and the cost of each one. Write a program that initializes the file "hardware.txt" to 10 empty records, lets you input the data concerning each tool, enables you to list all your tools, lets you delete a record for a tool that you no longer have and lets you update any information in the file. The tool identification number should be the

record number. Use the following information to start your file:

Record #	Tool name	Quantity	Cost
3	Electric sander	7	57.98
17	Hammer	76	11.99
24	Jig saw	21	11.00
39	Lawn mower	3	79.50
56	Power saw	18	99.99
68	Screwdriver	106	6.99
77	Sledge hammer	11	21.50
83	Wrench	34	7.50

4. Using C, create a file named budge.txt that contains three equal-length columns of numbers, like this:

```
 -462.13  486.47    973.79
755.42    843.04    -963.67
442.58   -843.02    -462.86
-233.93  -821.67    399.59
-379.65  -556.37    837.46
55.18     -144.93   -93.15
533.73    804.64    -66.25
-922.12   914.68    -264.67
-600.27  -838.59    747.02
-962.97   49.96     -677.79
```

Now write a program named budget.c that reads this file and adds up the numbers in each column. The program's output should look like this:

Column sums are: -1774.16 -105.79 429.47

5. Write a C++ Program to Count Digits, Alphabets and Spaces using File Handling.

Lab 13 Evaluation		
Student Name:	**Student ID:**	**Date:**
Task No.	**Marks**	**Remarks by teacher in accordance with the rubrics**
1		
2		
3		
4		
5		