Binary File

A binary file is a collection of bytes. A very easy way to find out whether a file is a text file or a binary file is to open that file in Turbo C/C++. If on opening the file you can make out what is displayed then it is a text file, otherwise it is a binary file. A binary file stores data into blocks of bytes. These blocks can be more complex data structures, such as arrays and structures. A set of library functions is available for processing binary files. Using these library functions we can transfer entire arrays or structures to or from data files using single statement. Binary files are more complicated to work with for certain kinds of applications.

6.7.4 File Pointer

The FILE structure is defined in the header file stdio.h. When working with a stream-oriented data file, the first step is to establish a buffer area, where information is temporarily stored while being transferred between the computer's memory and the data file. This buffer area allows information to be read from or written to the data file more rapidly than would otherwise be possible. A pointer of type FILE is called a filepointer, which references a disk file. A file pointer is used by a stream to conduct the operation of the I/O functions.For instance, the following defines a file pointer called fptr:

```
FILE *fptr;
```

where FILE (uppercase letters required) is a special structure type that creates the buffer area, and fptr is a pointer variable that indicates the beginning of the buffer area. In the FILE structure there is a member, called the file position indicator that points to the position in a file wheredata will be read from or written to.

6.7.5 File Operations

Following operations can be performed on files:

- Creation of a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Moving to a specific location in a file (seeking)
- Closing a file

6.8 BASIC FILE OPERATIONS ON TEXT FILES

(1) Creating an empty file.

(2) Opening a file – A file is opened for reading/writing or manipulation of data on it.

(3) Writing text into file – Once a file is created, data elements can be stored to it permanently.

(4) Reading of text from an already existing text file (accessing sequentially)

(5) Manipulation of text file from an already existing file. For example – counting of words.

(6) Copying of one text file to other text file

(7) Closing a file – After the file operations done, the file should be closed.

6.8.1 Opening and Closing Files in Text mode

Opening a File

Before we can read (or write) information from (to) a file on a disk we must open the file. To open the file fopen( )function is used.

This function is typically written as

```
fp = fopen( file-name, file-type);
```

- file-name represent the name of the data file
- file- type is the file opening mode that shows the manner in which the data file will be utilized i.e., as a read-only file, a write-only file, or a read write file.

File Opening Modes

When using fopen() function to open any file we have to specify the mode in which we want to open the file. There are two categories of file opening modes:

(1) Text files modes

(2) Binary files modes

Text Modes

The text file mode must be one of the following file opening modes. Read(r), write(w), append(a) modes

Mode	Meaning
"r"	Open an existing file for reading purpose only
"w"	Open a new file for writing only. If file exists overwrite it.
"a"	Open file for adding new data at the end of file. New file will be created if it doesn't exists
"r+"	Open an existing file for both reading and writing.
"w+"	Open an existing file for both reading and writing. Existing file will be overwritten. New file will be created if no file exists.
"a+"	Open an existing file for both reading and appending. New file will be created if no file exists.

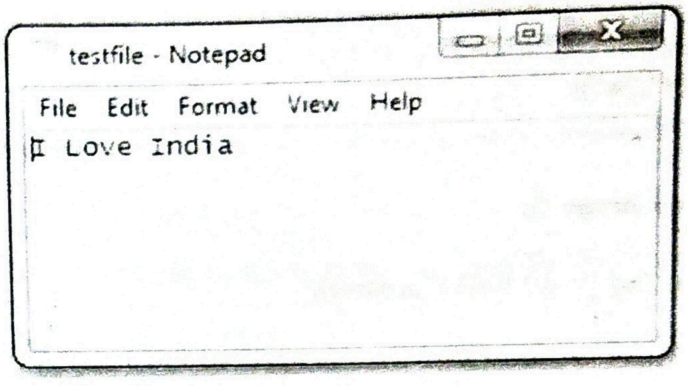
Example program:

Start with writing small program in text modes:

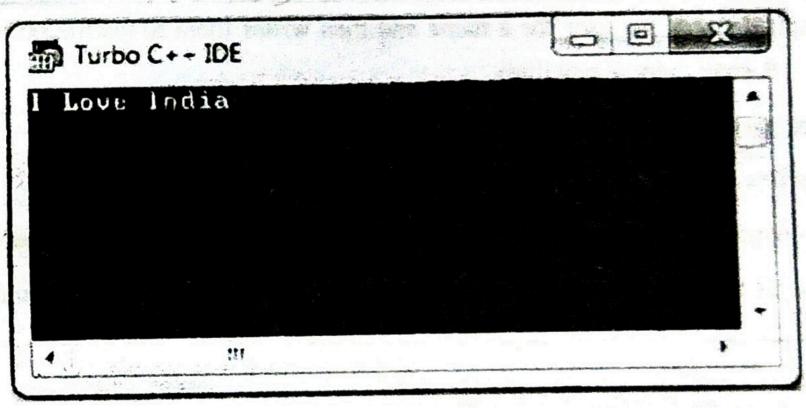
```
/* Display contents of a file on screen. */
#include <stdio.h>
main( )
{
FILE *fp ;
char ch ;
fp = fopen ( "testfile.txt", "r" ) ;
while ( 1 )
{
```

```
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
printf ( "%c", ch ) ;
}
fclose ( fp ) ;
}
```

Create a testfile.txt file inside the TC->BIN directory as follows:



On execution of this program, it displays the contents of the file 'testfile.txt on the screen.



6.8.2 Reading and Writing Files in Text mode

fprintf()

The fprintf function writes formatted output to stream.The syntax for the fprintf function is:

```
int fprintf(FILE *stream, const char *format, ...);
```

- stream is the stream where the output will be written.
- format describes the output as well as provides a placeholder to insert the formatted string. Here are a few examples:

Example

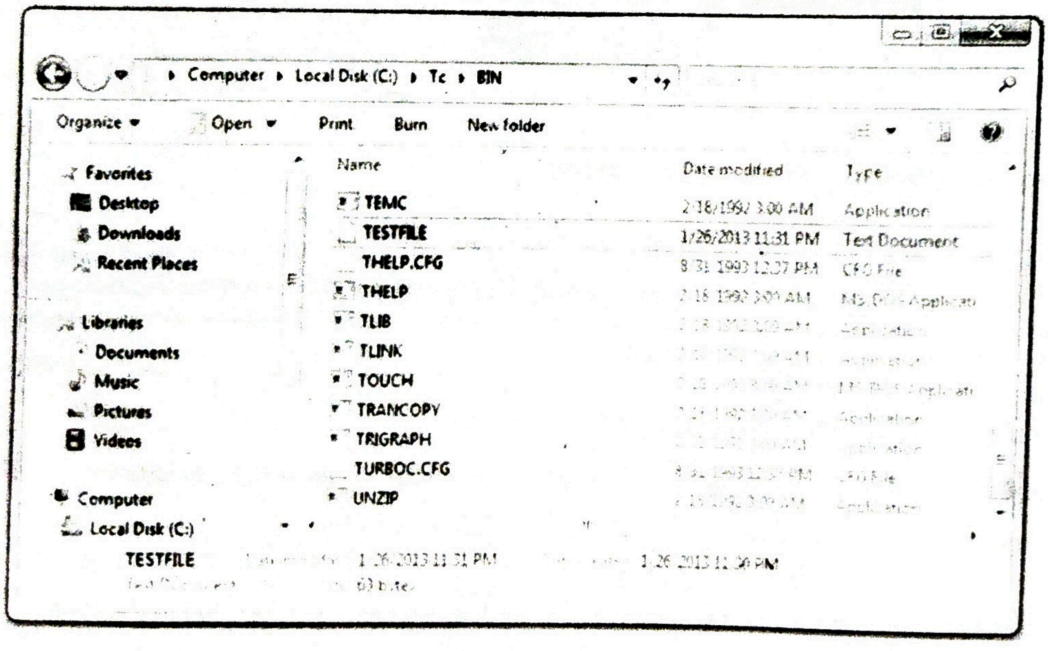
```
/* fprintf example */
#include <stdio.h>

int main ()
{
  FILE * fptr;
  int n;
  char name [100];

fptr = fopen ("testfile.txt","w");
  for (n=0 ; n<3 ; n++)
  {
    puts ("please, enter a name: ");
    gets (name);
    fprintf (fptr, "Name %d [%-10.10s]\n",n,name);
  }
  fclose (fptr);

  return 0;
}
```

This example prompts 3 times the user for a name and then writes them to testfile.txt each one in a line with a fixed length (a total of 19 characters + newline).

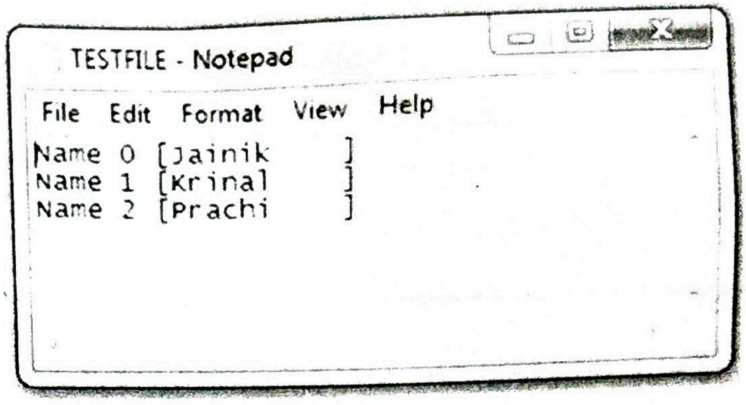
Two format tags are used:

- %d : Signed decimal integer
- %-10.10s : left-justified (-), minimum of ten characters (10), maximum of ten characters (.10), string (s).

By default the testfile.txt file will be created in the current directory from where C is running. In my case it is here:



Assuming that we have entered Jainik, Krinal and Prachi as the 3 names, testfile.txt would contain following information:



fscanf()

The fscanf function reads formatted output from stream.The syntax for the fscanf function is:

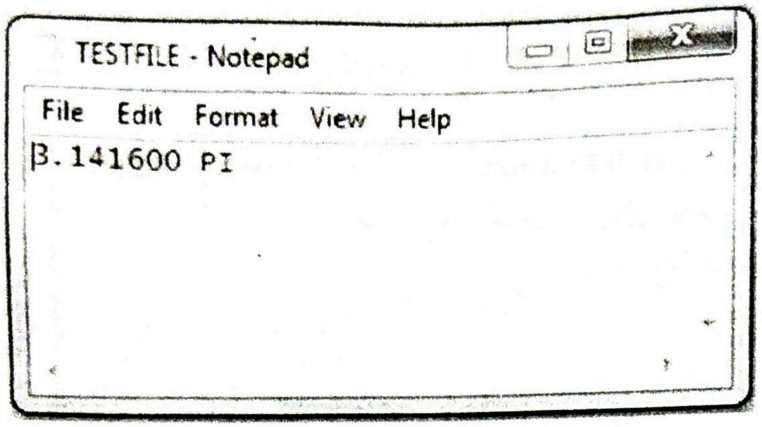
```
int fscanf(FILE *stream, const char *format, ...);
```

- stream is the stream where the output will be written.
- format describes the output as well as provides a placeholder to insert the formatted string. Here are a few examples:

Example

```
/* fscanf example */
#include <stdio.h>
int main ()
{
  char str [80];
  float f;
  FILE * fptr;

 fptr = fopen ("testfile.txt","w+");
  fprintf (fptr, "%f %s", 3.1416, "PI");
  rewind (fptr);
  fscanf (fptr, "%f", &f);
  fscanf (fptr, "%s", str);
  fclose (fptr);
  printf ("I have read: %f and %s \n",f,str);
  return 0;
 }
```

This sample code creates a file called testfile.txt and writes a float number and a string to it.



Then, the stream is rewind and both values are read with fscanf. It finally produces an output similar to:

```
I have read: 3.141600 and PI
```

EXERCISE

■ MCQ

1. In C, a Union is

   (a) memory location (b) memory store (c) memory screen (d) None of these

   Ans. : (b)

2. A Structure

   (a) can be read as a single entity (b) cannot be read as a single entity

   (c) can be displayed as a single entity (d) has member variables that cannot be read individually

   Ans. : (b)

3. Which of the following are themselves a collection of different data types?

   (a) String (b) Structures (c) Char (d) None of the above

   Ans. : (b)

4. Which of the following cannot be a structure member?

   (a) Function (b) Array (c) Structure (d) None of the above

   Ans. : (a)

5. Union differs from structure in the following way

   (a) All members are used at a time (b) Only one member can be used at a time

   (c) Union cannot have more members (d) Union initialized all members as structure

   Ans. : (b)

6. size of union is size of the longest element in the union

   (a) Yes (b) No (c) May Be (d) Can't Say

   Ans. : (a)

Questions

1. Define a structure called *record* which holds an integer called *loop*, a character array of 5 elements called *word*, and a float called *sum*.
2. Declare a structure variable called *sample*, defined from a structure of type *record*.
3. Assign the value 10 to the field *loop* of the *sample* structure of type record.
4. Print out (using printf) the value of the word array of the *sample* structure.
5. Define a new structure called *birthdays*, whose fields are a structure of type *time* called *btime*, and a structure of type *date*, called *bdate*.
6. Declare a pointer to a structure of type *date* called *dates*.
7. If the above structure of type date comprises three integer fields, day, month, year, assign the value 10 to the field *day* using the *dates* pointer.
8. A structure of type *machine* contains two fields, an integer called *name*, and a char pointer called *memory*. Show what the definition of the structure looks like.
9. A structure pointer *times* of type *time* (which has three fields, all pointers to integers, day, month and year respectively) is declared. Using the pointer *times*, update the field *day* to 10.
10. An array of pointers (10 elements) of type *time* (as detailed above in ' 7.), called *sample* is declared. Update the field *month* of the third array element to 12.
11. What is file? Give types of Files used in C?
12. Explain various file handling operations in C giving example.
13. List various file opening modes available in C.

Programming Exercises

1. Define a structure called record which holds an integer called loop, a character array of 5 elements called word, and a float called sum.
2. Declare a structure variable called sample, defined from a structure of type record.
3. Assign the value 10 to the field loop of the sample structure of type record.
4. Print out (using printf) the value of the word array of the sample structure.
5. Write a program that writes your name and student number and subject to a disk file. you name student number and subject name should be saved on new lines.
6. Read the file one character at a time and write it to screen
7. Write to the file again appending the date.
8. Read the file and output only the numeric information to screen