© November 2015 | IJIRT | Volume 2 Issue 6 | ISSN: 2349-6002

FILE MANAGEMENT IN C

Ria Arora
Computer Science and Engineering Department
Dronacharya College of Engineering,
Gurgaon, Haryana

ABSTRACT:- **: In real life we deal with large volume of data and therefore the concept of file management is necessary. Dealing with a huge amount of data is very cumbersome and time consuming. If we turn off the computer or if the program is terminated incorrectly, the entire data is lost.**
Due to these major problems it was very much essential to rely on file management where the data can be stored on the discs and read whenever required. Also, this does not destroy the data in any way.

File management in C involves the following:

- **Naming a file**
- **Opening a file**
- **Writing a file**
- **Reading a file**
- **Closing a file**

INTRODUCTION

What is basically a file? It is a collection of bytes on secondary storage device. This storage device is generally a disc of some kind. In a file, the collection of bytes are sometimes interpreted, like, characters, words, line, paragraphs and pages from a document. Also, fields and reports of a database may mingle or it may simply be the pixels of a graphical image. The data structures and the operations used by a program to process the file determine meaning attached to a particular file. Sometimes, the program designed to process textual data reads and displays a graphics file. As a result, a meaningless output is what we get. This is not what the user expects. In a file programs and data are stored for machine usage as it is simply a machine decipherable storage media.

OVERVIEW

A programmer deals with two types of files:
- Text files
- Binary files

BINARY FILES
A Binary file is similar to a text file. This file is a collection of bytes. A byte and a character are equivalent in a C programming file. A character stream is what a file is referred to as. But, there are two main differences:
- The data is not processed in any special way and every byte of data is transferred to or from the disc unprocessed.
- The file can be read or written in any manner desired by the programmer as the C language has no restrictions regarding this. Depending on the needs of the application, binary files can be either processed sequentially or, they can be processed using random access techniques. Moving the current file position to an appropriate place in the file before reading or writing data in C Programming Language, is done by processing a file using random access techniques. This is the second characteristic of binary files.

After opening a binary file, the user can seek a specific position in the file or read and write a structure of the file. When the file is opened, a file position indicator points to record 0. The file position indicator points at a specific position and at this position the read operation reads the structure. The pointer is moved to point at the next structure after reading it. The currently pointed structure allows to write at its location using write operation. The file position indicator is moved to point at the next structure after the user is done with the write operation. The file position indicator is moved to the record that is requested by the seek function.

The user need to remember to keep track of things, because not only the beginning of a structure, but can also any byte in the file can be pointed by the position indicator.
Four parameters are taken care of by the fread and fwrite functions:
- A memory address
- Number of bytes to read per block
- Number of blocks to read

- A file variable

ASCII TEXT FILES

A computer can process sequentially a text file which is considered to be a stream of characters. It is processed sequentially as well as in forward direction. At any given time a text file is generally opened for only one kind of operation (reading, writing, or appending).

In the same way, only one character at a time can be read by the text files since they only process characters. A special kind of file is a text stream in C. Depending on whether data is being written to, or read from, the file newline characters may be converted to or from carriage-return/linefeed combinations as per the needs of the operating system. To satisfy the storage requirements of the operating system other character conversions may also occur. These translations occur because the programmer has signaled the intention to process a text file and they occur transparently.

METHODOLOGY

Console oriented Input/Output

Console oriented input/output makes use of terminals like keyboard and screen. The command scanf is used to take input from user through keyboard and the printf command is used to display output on the screen. It is suitable for small amount of data. Also, the data is lost when program is terminated.

Real Life Applications

Large data input is what we come across in real life. For example, physical experiments, human genome, population records, academic records etc. There is need for flexible approach to store and retrieve data. Therefore the concept of files evolved.

Files

File is a place on disc where a group of related data is stored. High level programming languages which support file operations are:

- Naming
- Opening
- Reading
- Writing
- Closing

Defining and Opening a File

To store data file in secondary memory (disc) must specify to the operating system.

- Filename (e.g. sort.c, input.data)
- Data structure (e.g. FILE)
- Purpose (e.g. reading, writing, appending)

Filename

A string of characters make up a valid filename for the operating system. It may contain two part namely: Primary and optional period with extension. For example: prog.c, a.out, temp, text.out etc.

General Format for Opening a File

FILE *fp; /*variable fp is pointer to type FILE*/
fp = fopen(“*filename*”, “*mode*”);
/*opens file with name *filename* , assigns identifier to fp */

fp:

- It contains all the information about the file.
- It also serves as a communication link between system and program.

Mode can be:

- r- open file for reading only
- w- open file for writing only
- a- open a file for appending(adding) data

Different Modes

I. Writing Mode
    - If file already exists then contents are deleted.
    - Else a new file with specific name is created

II. Appending mode
    - If file already exists then file opened with contents safe
    - Else new file created

III. Reading mode
    - If file already exists then opened with contents safe
    - Else error occurs.

Closing a File

The file must be closed as soon as all operations on it are completed. It ensures that-

- All outstanding information associated with file flushed out from buffers
- All links to file broken
- Accidental misuse of file prevented
- For changing the mode of file, first close it and then open again. Also, the pointer can be reused after closing

Syntax: **fclose**(file_pointer);
Example:
FILE *p1, *p2;
p1 = fopen("INPUT.txt", "r");
p2 =fopen("OUTPUT.txt", "w");
fclose(p1);
fclose(p2);

<u>File Management functions in C:</u>

- getc()-to read a character from a file
- putc()-to write a character to a file
- fopen()-to create a new file for use
- fclose()-to close a file which was opened for use
- getw()-to read an integer from a file
- putw()-to write an integer to a file
- fprintf()-to write a set of data values from a file
- fscanf()-to read a set of data values from a file
- rewind()-to set the position to the beginning of the file
- fseek()-to set the position at a desired point in the file
- ftell()-to give the current position in the file

<u>Errors that occur during I/O Operations</u>
Typical errors that occur are:

- Trying to read beyond end-of-file
- Trying to use a file that has not been opened
- Perform operation on file not permitted by 'fopen' mode
- Open file with invalid filename
- Write to write-protected file

<u>Error handling</u>

- given file-pointer, check if EOF reached, errors while handling file, problems opening file etc.
- check if EOF reached: feof()
- feof() takes file-pointer as input, returns nonzero if all data read and zero otherwise

if(feof(fp))
printf("End of data\n");

- ferror() takes file-pointer as input, returns nonzero integer if error detected else returns zero
- if(ferror(fp) !=0)
- printf("An error has occurred\n");

CONCLUSION

- File management is the new way of dealing with large amount of data.
- The data can be retrieve whenever require as it is stored on the disk.
- The disk may store output of the data.
- We have a wide range of functions in C which deal with file handling.

REFERENCES

- https://en.wikipedia.org/wiki/C_file_input/output
- http://www.thegeekstuff.com/2012/07/c-file-handling/
- http://www.sanfoundry.com/c-programming-examples-file-handling/
- http://www.peoi.org/Courses/Coursesen/cprog/frame13.html
- http://www.tutorialspoint.com/cprogramming/c_file_io.htm
- http://www.dailyfreecode.com/code/file-management-2767.aspx