Files in C

File handling in C is the process in which we create, open, read, write, and close operations on a file. C language provides different functions such as fopen(), fwrite(), fread(), fseek(), fprintf(), etc. to perform input, output, and many different C file operations in a program.

Why is File Handling needed in C?

So far the operations using the C program are done on a prompt/terminal which is not stored anywhere. The output is deleted when the program is closed. But in the software industry, most programs are written to store the information fetched from the program. The use of file handling is exactly what the situation calls for.

In order to understand why file handling is important, let us look at a few features of using files:

- **Reusability:** The data stored in the file can be accessed, updated, and deleted anywhere and anytime providing high reusability.
- **Portability:** Without losing any data, files can be transferred to another in the computer system. The risk of flawed coding is minimized with this feature.
- **Efficient:** A large amount of input may be required for some programs. File handling allows you to easily access a part of a file using few instructions which saves a lot of time and reduces the chance of errors.
- **Storage Capacity:** Files allow you to store a large amount of data without having to worry about storing everything simultaneously in a program.

Types of Files in C

A file can be classified into two types based on the way the file stores the data. They are as follows:

- **Text Files**
- **Binary Files**

1. Text Files

A text file contains data in the **form of ASCII characters** and is generally used to store a stream of characters.

- Each line in a text file ends with a new line character (‘\n’).
- It can be read or written by any text editor.
- They are generally stored with **.txt** file extension.
- Text files can also be used to store the source code.

2. Binary Files

A binary file contains data in **binary form (i.e. 0’s and 1’s)** instead of ASCII characters. They contain data that is stored in a similar manner to how it is stored in the main memory.

- The binary files can be created only from within a program and their contents can only be read by a program.
- More secure as they are not easily readable.
- They are generally stored with **.bin** file extension.

C File Operations

C file operations refer to the different possible operations that we can perform on a file in C such as:

1. Creating a new file – **fopen() with attributes as "a" or "a+" or "w" or "w+"**
2. Opening an existing file – **fopen()**
3. Reading from file – **fscanf() or fgets()**
4. Writing to a file – **fprintf() or fputs()**
5. Moving to a specific location in a file – **fseek(), rewind()**
6. Closing a file – **fclose()**

Functions for C File Operations

File operation	Declaration & Description
fopen() - To open a file	Declaration: FILE *fopen (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=fopen ("filename", "'mode"); Where, fp – file pointer to the data type "FILE". filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.
fclose() - To close a file	Declaration: int fclose(FILE *fp); fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below. fclose (fp);
fgets() - To read a file	Declaration: char *fgets(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. fgets (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer
fprintf() - To write into a file	Declaration: int fprintf(FILE *fp, const char *format, ...);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or fprintf (fp, "text %d", variable_name);

File Pointer in C

A file pointer is a reference to a particular position in the opened file. It is used in file handling to perform all file operations such as read, write, close, etc. We use the **FILE** macro to declare the file pointer variable. The FILE macro is defined inside **<stdio.h>** header file.

Syntax of File Pointer

FILE* *pointer_name*;

Open a File in C

For opening a file in C, the fopen() function is used with the filename or file path along with the required access modes.

Syntax of fopen()

FILE* **fopen**(const char **file_name*, const char **access_mode*);

Parameters

- *file_name:* name of the file when present in the same directory as the source file. Otherwise, full path.
- *access_mode:* Specifies for what operation the file is being opened.

Return Value

- If the file is opened successfully, returns a file pointer to it.
- If the file is not opened, then returns NULL.

File opening modes in C

File opening modes or access modes specify the allowed operations on the file to be opened. They are passed as an argument to the fopen() function. Some of the commonly used file access modes are listed below:

Opening Modes	Description
r	Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
rb	Open for reading in binary mode. If the file does not exist, fopen( ) returns NULL.
w	Open for writing in text mode. If the file exists, its contents are overwritten. If the file doesn't exist, a new file is created. Returns NULL, if unable to open the file.
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	Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. It opens only in the append mode. If the file doesn't exist, a new file is created. Returns NULL, if

Opening Modes	Description
	unable to open the file.
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+	Searches file. It is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the first character in it. Returns NULL, if unable to open the file.
rb+	Open for both reading and writing in binary mode. If the file does not exist, fopen( ) returns NULL.
w+	Searches file. If the file exists, its contents are overwritten. If the file doesn't exist a new file is created. Returns NULL, if unable to open the file.
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+	Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. It opens the file in both reading and append mode. If the file doesn't exist, a new file is created. Returns NULL, if unable to open the file.
ab+	Open for both reading and appending in binary mode. If the file does not exist, it will be created.

Creating a File in C

The fopen() function can not only open a file but also can create a file if it does not exist already. For that, we have to use the modes that allow the creation of a file if not found such as w, w+, wb, wb+, a, a+, ab, and ab+.

```
FILE *fptr;
fptr = fopen("filename.txt", "w");
```

Reading From a File

The file read operation in C can be performed using functions fscanf() or fgets(). Both the functions performed the same operations as that of scanf and gets but with an additional

parameter, the file pointer. There are also other functions we can use to read from a file. Such functions are listed below:

Function	Description
fscanf()	Use formatted string and variable arguments list to take input from a file.
fgets()	Input the whole line from the file.
fgetc()	Reads a single character from the file.
fgetw()	Reads a number from a file.
fread()	Reads the specified bytes of data from a binary file.

Example:

```
FILE * fptr;
fptr = fopen(“fileName.txt”, “r”);
fscanf(fptr, "%s %s %s %d", str1, str2, str3, &year);
char c = fgetc(fptr);
```

The getc() and some other file reading functions return EOF (End Of File) when they reach the end of the file while reading. EOF indicates the end of the file and its value is implementation-defined.

Writing to a File

The file write operations can be performed by the functions fprintf() and fputs() with similarities to read operations. C programming also provides some other functions that can be used to write data to a file such as:

Function	Description
fprintf()	Similar to printf(), this function use formatted string and varible arguments list to print output to the file.

Function	Description
fputs()	Prints the whole line in the file and a newline at the end.
fputc()	Prints a single character into the file.
fputw()	Prints a number to the file.
fwrite()	This functions write the specified amount of bytes to the binary file.

Example:

```
FILE *fptr ;
fptr = fopen(“fileName.txt”, “w”);
fprintf(fptr, "%s %s %s %d", "We", "are", "in", 2012);
fputc("a", fptr);
```

Closing a File

The fclose() function is used to close the file. After successful file operations, one must always close a file to remove it from the memory.

Syntax of fclose()

```
fclose(file_pointer);
```

where the *file_pointer* is the pointer to the opened file.

Example:

```
FILE *fptr ;
fptr= fopen(“fileName.txt”, “w”);
---------- Some file Operations -------
fclose(fptr);
```

Read and Write in a Binary File

Till now, we have only discussed text file operations. The operations on a binary file are similar to text file operations with little difference.

Opening a Binary File

To open a file in binary mode, we use the rb, rb+, ab, ab+, wb, and wb+ access mode in the fopen() function. We also use the .bin file extension in the binary filename.

Example

```
fptr = fopen("filename.bin", "rb");
```

Write to a Binary File

We use fwrite() function to write data to a binary file. The data is written to the binary file in the from of bits (0's and 1's).

Syntax of fwrite()

size_t **fwrite**(const void *_ptr_, size_t _size_, size_t _nmemb_, FILE *_file_pointer_);

Parameters:

- **ptr:** pointer to the block of memory to be written.
- **size:** size of each element to be written (in bytes).
- **nmemb:** number of elements.
- **file_pointer**: FILE pointer to the output file stream.

Return Value:

- Number of objects written.

Reading from Binary File

The fread() function can be used to read data from a binary file in C. The data is read from the file in the same form as it is stored i.e. binary form.

Syntax of fread()

size_t **fread**(void *_ptr_, size_t _size_, size_t _nmemb_, FILE *_file_pointer_);
Parameters:

- **ptr:** pointer to the block of memory to read.
- **size:** the size of each element to read(in bytes).
- **nmemb:** number of elements.
- **file_pointer**: FILE pointer to the input file stream.

Return Value:

- Number of objects written.

fseek() in C

If we have multiple records inside a file and need to access a particular record that is at a specific position, so we need to loop through all the records before it to get the record. Doing this will waste a lot of memory and operational time. To reduce memory

consumption and operational time we can use fseek() which provides an easier way to get to the required data. fseek() function in C seeks the cursor to the given record in the file.

Syntax for fseek()

```
int fseek(FILE *ptr, long int offset, int pos);
```

rewind() in C

The rewind() function is used to bring the file pointer to the beginning of the file. It can be used in place of fseek() when you want the file pointer at the start.

Syntax of rewind()

```
rewind (file_pointer);
```

More Functions for C File Operations

The following table lists some more functions that can be used to perform file operations or assist in performing them.

Functions	Description
fopen()	It is used to create a file or to open a file.
fclose()	It is used to close a file.
fgets()	It is used to read a file.
fprintf()	It is used to write blocks of data into a file.
fscanf()	It is used to read blocks of data from a file.
getc()	It is used to read a single character to a file.
putc()	It is used to write a single character to a file.

Functions	Description
fseek()	It is used to set the position of a file pointer to a mentioned location.
ftell()	It is used to return the current position of a file pointer.
rewind()	It is used to set the file pointer to the beginning of a file.
putw()	It is used to write an integer to a file.
getw()	It is used to read an integer from a file.