Basics of File Handling 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 our program.

Why do we need File Handling 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()**

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;
```

- File Pointer is used in almost all the file operations in C.

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.

Example of Opening a File:

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

int main()
{
        FILE * fptr;

        fptr = fopen("file_name.txt", "r");

        if(fptr == NULL)
        {
                printf("The file is not opened. The program will " "now exit.\n");
                exit(0);

        }

        return 0;

}
```

OUTPUT:

The file is not opened. The program will now exit.

Create 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.

Syntax:

FILE *fptr;
fptr = fopen(‘’filename.txt”,”w”);

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

int main()
{
    // file pointer
    FILE* fptr;

    // creating file using fopen() access mode "w"
    fptr = fopen("file.txt", "w");

    // checking if the file is created
    if (fptr == NULL) {
        printf("The file is not opened. The program will " "exit now\n");
        exit(0);
    }
    else {
        printf("The file is created Successfully.\n");
    }

    return 0;
}
```

Output:

The file is created Successfully.

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.

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() **fread()**	Reads a number from a file. Reads the specified bytes of data from a binary file.

Write to a File

The file write operations can be performed by the functions fprintf() and fputs() with similarities to read operations.

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

Closing a File

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

Syntax of fclose():

- fclose(file_pointer);

Example 1: Program to Create a File, Write in it, And Close the File

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

int main()
{

        FILE *filepointer;
        char data_to_be_written[50] = "Ramakrishna KPIT";

        filepointer = fopen("write_file.c","w");

        if(filepointer == NULL)
        {
                printf("write_file.c failed to open\n");

        }
        else
        {
                printf("The file is now opend \n");

                if(strlen(data_to_be_written)> 0)
                {
                        fputs(data_to_be_written,filepointer);//writing in the file using fputs()

                        fputs("\n",filepointer);
                }

                fclose(filepointer);



        printf("Data successfully written in file , write_file.c\n");
        printf("the file is closed\n");
    }
        return 0;
}
```

Output:

The file is now opend
Data successfully written in file , write_file.c
the file is closed

Example 2: Program to Open a File, Read from it, And Close the File

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

int main()
{

    FILE* filePointer;

    char dataToBeRead[50];

    filePointer = fopen("write_file.c", "r");

    if (filePointer == NULL) {
        printf("write_file.c file failed to open.");
    }
    else {

        printf("The file is now opened.\n");

        while (fgets(dataToBeRead, 50, filePointer)!= NULL)
        {


            printf("%s", dataToBeRead);
        }
        fclose(filePointer);

        printf("Data successfully read from file GfgTest.c\n");
        printf("The file is now closed.");
    }
    return 0;
}
```

Output:
The file is now opened.
Ramakrishna KPIT
Data successfully read from file GfgTest.c
The file is now closed.