File Handling in C

"File handling in C" helps us create, manipulate, update, and delete files stored on the local file system using C programs.

File Handling in C Definition: File Handling in C programming language is a feature that lets you operate the files with the help of C programs or C code.

Sometimes merely displaying the facts on the console is insufficient. The quantity of data displayed on the console may be extremely small or very vast. Because the memory is volatile, it is hard to repeatedly restore the programmatically generated data. However, we can store data on the local file system, which is volatile and accessible whenever we need to. Here, file handling in C becomes necessary.

What is a File in C?

A file in C can be defined as a sequence of bytes. A file can be a text file straight away or a binary code sequence. Regardless of this, it is treated as a sequence of bytes. It can be a text file, binary file, or any other type of file. A file can contain various types of data, such as numbers, characters, strings, or images.

In C, a file is represented by a pointer to a structure called "FILE". The C programming language for managing files on storage devices offers access to both low-level (OS level) calls and high-level functions.

Types of Files in C

A file is typically used in computers to store user data. In other words, a computer uses files to store data. These data files are now available in the C language in 2 different forms, namely:

- Binary Files
- Text Files

Now we shall briefly discuss each type of file in C.

Binary File Handling in C

The binary files contain information and data using the 0s and 1s binary coding scheme (the binary number system). As a result, the files take up far less storage space. In other words, binary files retain data and information like a computer does in its memory. As a result, it is easier to access than a text file.

The disadvantage of text files in a program is overcome by the binary files, which are produced with the extension.bin and can only be read by computers, as opposed to humans, who cannot read them.

Text File Handling in C

The most fundamental types of files that a user in a C program can produce are text files. We produce the text files using a basic text editor with the.txt extension. These files internally hold information in ASCII character format. However, the content or text appears in a readable format when we view them.

Thus, accessing and using text files is quite simple. There is, however, one significant drawback: lack of security. Information isn't secure in a text file because it can be retrieved quickly. Apart from the security issue, text files occupy a larger storage space than binary files.

Functions for File Handling in C

A file can be opened, read, expanded upon, created, closed, deleted, searched for, etc., via several different functions. In C, these are referred to as file handling operators.

The following is a list of the functions that let you do that:

Description of Function	Function in Use
To open an existing file or a new file	fopen()
For writing data into a file	fprintf()
For reading the data in a file	fscanf()
For writing any character into the program file	fputc()
For reading the character from a file	fgetc()
To close the program file	fclose()
To set the file pointer to the intended file position	fseek()
For writing an integer into an available file	fputw()
To read an integer from the given file	fgetw()
For reading the current position of a file	ftell()
To set an intended file pointer to the file's beginning itself	rewind()

Operations in File Handling in C

A user can update, create, open, read, write, and finally delete the file/content in the file that already exists on the local file system of the C program through the file handling process. The main file operations you can carry out in a C program are listed below:

- Opening a file
- Creating a new file
- Writing data to file
- Reading data from file
- Deleting data from file

Need for File Handling in C

Occasionally, a program's output after it has been compiled and run does not meet our objectives. We might want to check the program's output several times in these circumstances. It takes a lot of work for any coder to repeatedly compile and run the same software. This is the precise situation when file handling is helpful. Some of the features of file handling in C that provide a user or programmer with convenience are:

- Portability of the source code.
- Provides storage capacity.
- Provides reusability of the programs that are stored in the file.

Error Handling in File Operations in C

In C programming, it is important to handle errors that may occur during file operations. Errors can occur due to various reasons such as file not found, insufficient permissions, etc. To handle errors, C provides the errno variable, which is a global integer variable that contains the error code of the last file operation that failed.In C programming, it is important to handle errors that may occur during file operations. Errors can occur due to various reasons such as file not found, insufficient permissions, etc. To handle errors, C provides the errno variable, which is a global integer variable that contains the error code of the last file operation that failed.
In addition to errno, C also provides the perror() function, which prints the error message corresponding to the value of errno. The perror() function takes a string argument that is used as a prefix for the error message.
Here is an example of error handling in file operations in C:

```
#include <stdio.h>#include <stdio.h>#include <errno.h>
int main() { FILE *fp;
fp = fopen("non_existent_file.txt", "r"); if(fp == NULL) { perror("Error"); printf("errno =
%d\n", errno); }
return 0;}
```

In this example, we are trying to open a file that does not exist. The fopen() function returns a NULL pointer, indicating that the file could not be opened. We then use the perror() function to print the error message along with the value of errno.In this example, we are trying to open a file that does not exist. The fopen() function returns a NULL pointer, indicating that the file could not be opened. We then use the perror() function to print the error message along with the value of errno.

By handling errors in this way, we can ensure that our program does not crash due to file operations that fail.

Examples of File Handling in C Programming

Examples of File Handling in C Programming can include:

Action	Description
Reading a file	Opening an existing file and reading its contents.
Writing to a file	Creating a new file or overwriting an existing file and writing data to it.
Appending to a file	Opening an existing file and adding data to its end.
Closing a file	Terminating the connection between the program and the file.
Checking for errors	Ensuring that the file operations are successful and handling errors if they occur.