C File Handling

A file is a container in computer storage devices used for **storing data.**

A **file** represents a **sequence of bytes on the disk** where a **group of related data is stored**.

File is created for **permanent storage of data**.

It is a **readymade structure.**

Why files are needed?

- When a program is terminated, the entire data is lost. **Storing in a file will preserve your data** even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all.
  However, if you have a file containing all the data, **you can easily access the contents of the file** using a few commands in C.
- You can **easily move your data from one computer to another without** any changes.

Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

1. Text files

- Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.
- When you open those files, you'll see all the contents within the file as plain text.
- You can **easily edit or delete** the contents.
- They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

2. Binary files

- Binary files are mostly the **.bin** files in your computer.
- Instead of storing data in plain text, they store it in the **binary form (0's and 1's).**
- They can hold **a higher amount of data**, are **not readable** easily, and provides **better security** than text files.

File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Function	description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

In C language, we use a structure pointer of file type to declare a file.

```
FILE *fp;
```

Opening a File or Creating a File

The fopen() function is used to create a new file or to open an existing file.

General Syntax:

*fp = FILE *fopen(const char *filename, const char *mode);

Here, *fp is the FILE pointer (FILE *fp), which will hold the reference to the opened(or created) file.

filename is the name of the file to be opened and **mode** specifies the purpose of opening the file. Mode can be of following types,

MODE	DESCRIPTION
R	opens a text file in reading mode
W	opens or create a text file in writing mode.
A	opens a text file in append mode
R+	opens a text file in both reading and writing mode
W+	opens a text file in both reading and writing mode
A+	opens a text file in both reading and writing mode
RB	opens a binary file in reading mode
WB	opens or create a binary file in writing mode
AB	opens a binary file in append mode
RB+	opens a binary file in both reading and writing mode
WB+	opens a binary file in both reading and writing mode
AB+	opens a binary file in both reading and writing mode

Closing a File

The fclose() function is used to close an already opened file.

General Syntax:

```
int fclose( FILE *fp);
```

Here fclose() function closes the file and returns **zero** on success, or **EOF** if there is an error in closing the file. This **EOF** is a constant defined in the header file **stdio.h**.

Input/Output operation on File

In the above table we have discussed about various file I/O functions to perform reading and writing on file. `getc()` and `putc()` are the simplest functions which can be used to read and write individual characters to a file.

```
#include<stdio.h>


int main()

{

    FILE *fp;

    char ch;

    fp = fopen("one.txt", "w");

    printf("Enter data...");

    while( (ch = getchar()) != EOF) {

        putc(ch, fp);

    }

    fclose(fp);

    fp = fopen("one.txt", "r");


    while( (ch = getc(fp)! = EOF)

    printf("%c",ch);


    // closing the file pointer

    fclose(fp);


    return 0;

}
```

Reading and Writing to File using fprintf() and fscanf()

```
#include<stdio.h>


struct emp

{

    char name[10];

    int age;

};


void main()

{

    struct emp e;

    FILE *p,*q;

    p = fopen("one.txt", "a");

    q = fopen("one.txt", "r");

    printf("Enter Name and Age:");

    scanf("%s %d", e.name, &e.age);

    fprintf(p,"%s %d", e.name, e.age);

    fclose(p);

    do

    {

        fscanf(q,"%s %d", e.name, e.age);

        printf("%s %d", e.name, e.age);

    }

    while(!feof(q));

}
```

In this program, we have created two FILE pointers and both are refering to the same file but in different modes. fprintf() function directly writes into the file, while fscanf() reads from the file, which can then be printed on the console using standard printf() function.