

File Handling in C Language

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.

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

```
FILE *fp;
```

C provides a number of functions that helps to perform basic file operations. Following are the functions,

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 beginning point

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 **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,

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

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 simplest functions used to read and write individual characters to a file.

```
#include<stdio.h>
#include<conio.h>
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()) != EOF)
    printf("%c",ch);
fclose(fp);
}
```

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

```
#include<stdio.h>
#include<conio.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) );
    getch();
}
```

In this program, we have create 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 console usinf standard **printf()** function.

Difference between Append and Write Mode

Write (w) mode and Append (a) mode, while opening a file are almost the same. Both are used to write in a file. In both the modes, new file is created if it doesn't exist already.

The only difference they have is, when you open a file in the write mode, the file is reset, resulting in deletion of any data already present in the file. While in append mode this will not happen. Append mode is used to append or add data to the existing data of file(if any). Hence, when you open a file in Append(a) mode, the cursor is positioned at the end of the present data in the file.

Reading and Writing in a Binary File

A Binary file is similar to the text file, but it contains only large numerical data. The Opening modes are mentioned in the table for opening modes above.

fread() and **fwrite()** functions are used to read and write in a binary file.

```
fwrite(data-element-to-be-written, size_of_elements,  
        number_of_elements, pointer-to-file);
```

fread() is also used in the same way, with the same arguments like fwrite() function. Below mentioned is a simple example of writing into a binary file

```
const char *mytext = "The quick brown fox jumps over the lazy  
dog";  
FILE *bfp= fopen("test.txt", "wb");  
if (bfp) {  
    fwrite(mytext, sizeof(char), strlen(mytext), bfp) ;  
    fclose(bfp) ;  
}
```

fseek(), ftell() and rewind() functions

- **fseek()** - It is used to move the reading control to different positions using fseek function.
 - **ftell()** - It tells the byte location of current position of cursor in file pointer.
 - **rewind()** - It moves the control to beginning of the file.
-