

## Module 10

### File Handling

With the help of these functions the user can open a file with the data file specifications, create and write information in to the file and can close the file. The following are the file processing related functions.

- a. FILE OPEN function `fopen()`
- b. FILE CLOSE function `fclose()`
- c. FILE INPUT functions `getc()` and `fscanf()`
- d. FILE OUTPUT functions `putc()` and `fprintf()`

#### a. The function `fopen()`

This function is used to open a data file. Moreover, this function returns a pointer to a file. The use of the function is

```
file_pointer = fopen( filename, mode);
```

Where file pointer is a pointer to a type FILE, the file name of the file name is name of the file in which data is stored or retrieved (should be enclosed within double quotes) and the mode denotes the type of operations to be performed on the file (this also to be enclosed within double quotes).

But, before making this assignment ,the file pointer and `fopen()` should be declared as FILE pointer type variables as under :

```
FILE *file_pointer, * fopen() ;
```

The mode can be one of the following types.

MODE	MEANING
"r"	read from the file
"w"	write to the file
"a"	append a file ie new data is added to the end of file
"r+"	open an existing file for the sake of updation.
"w+"	create a new file for reading and writing
"a+"	open a file for append, create a new one if the file does not exist already

#### Examples

1. `fptr = fopen("rk.txt", "w");`
2. `file= fopen("sample.txt", "r +");`

#### b.The functions `fclose ()`

The files that are opened should be closed after all the desired operations are performed on it .This can be achieved through this function .The usage of this function is:

```
fclose (file pointer);
```

Where file pointer is returned value of the `fopen ()` function.

#### Examples:

1. `fclose ( input file);`
2. `fclose (output file);`

#### c.The functions `getc()` & `fscanf()`

**1.getc ( ) functions:** this function is used a single character from a given file ,whenever a file is referenced by a file pointer. The usage of this function is  
`getc (file pointer);`

**2.fscanf ( )function :** This function is used to read a formatted data from a specified file. The general usage of this function is

```
fscanf (f ptr, "Control String", & list);
```

Where

Fptr = a file pointer to receive formatted data

Control string = data specifications list

List =the list of variables to be read

Example: `fscanf (infile , "%d %d , " & no, &marks);`

#### **d.The functions putc()& fprintf()**

**1.putc( ) function:** This function is used write a single character into a file referenced by the file pointer. The usage of this function is

putc (ch, file pointer),

Where

ch = the character to be written

file pointer = a file pointer to the file that receives the character.

**2.fprintf ( ) function:** this function is used to write a formatted data into a given file. The specified information is written on the specified file.

The general form of usage for this function is: fprintf (fptr, "Control String", list):

Where

Fptr file pointer to write a formatted data Control string data specifications list

list- list of variables to be written.

Example

fprintf (out file,"%d %f", basic , gross);

#### **C program to copy contents of one file to another file**

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;
    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);

    // Open one file for reading
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);

    // Open another file for writing
    fptr2 = fopen(filename, "w"); //opens a file if already present or creates a new file if file does not exist
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    // Read contents from file
    c = fgetc(fptr1); //reading from the file
    while (c != EOF)
    {
        fputc(c, fptr2); //writing to the file
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

## File Input/Output in C

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 ready made 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 begining 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, \*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 referring 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.

#### 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 a 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.