File Handling

File Handling is the storing of data in a **file** using a program.

File Handling concept in C language is used for store a data permanently in computer. Using this concept we can store our data in Secondary memory (Hard disk).

Various operations like opening the file reading/writing in a file and closing the file can be done.

Types of Files

There are two types of files

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 the data of a file is stored in the form of readable and printable characters then the file is known as a text file.

2. Binary Files

Binary files are mostly the **.bin** files in your computer.

If a file contains non-readable characters in binary code then the file is called a binary file.Instead of storing data in plain text, they store it in the binary form (0's and 1's).

Steps in processing a file

The processing of a file involves the following steps:

1. Opening a file:- before using any file, that file must be opened. So first step in file processing is to open the file.
If you want to read data from a file, the file must be opened in the input/read mode. Similarly, if you want to write the result of your program to me, it must be opened in output/write mode.
It may also be possible that you want to read data from and also write data onto the then the file must be opened in read-write mode.
2. Reading or writing a file:- it is the second step of processing the file. Once the file is opened successfully, data can be read to written into a file in various ways.
3. Closing the file:- it is the last and final step in processing file. Once we have completed the reading or/and writing of the each and every file must be closed.

Basic File Operations

There are four basic operations that can perform on any file in 'C'.

- Opening a file
- Closing a file
- Reading a file
- Writing a file

Functions in 'C' to perform Basic File Operations:

- **fopen()** The fopen() function is used to open a file.
- **fclose()**The fclose () function is used to closea file.
- **getc()** read 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.

1.Opening a file

Before using a data file in a program, we must have to open it. We must open a file before it can be read, write, or update. The fopen() function is used to open a file.
The general form of declaring and opening a file is: Declaring file pointer with

FILE *fp;

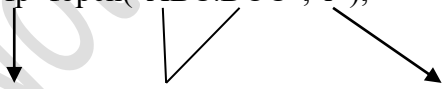
Where FILE is the data structure included in stdio.h and fp is file pointer which points to the start of the file.
Syntax:

FILE=*fp;

fp = fopen ("filename", "mode");

- Where fp is a pointer to the structure of type FILE, fopen( )is a library function having two arguments, one is filename and other is the mode in which we want to open the file.

Example: fp=fopen("ABC.DOC","r");



File pointer name of file mode in which file is opened.

Example: fp=fopen("c:\\programs\text.txt","r");

File Opening Modes

We can use one of the following modes in the fopen() function.

Mode	Description
R	opens a text file in read mode
W	opens a text file in write mode
A	opens a text file in append mode
r+	opens a text file in read and write mode
w+	opens a text file in read and write mode
a+	opens a text file in read and append mode
Rb	opens a binary file in read mode
Wb	opens a binary file in write mode
Ab	opens a binary file in append mode
rb+	opens a binary file in read and write mode.
wb+	opens a binary file in read and write mode
ab+	opens a binary file in read and append mode

2.Reading a File

For reading and writing to a text file, we use the some functions We can read content of a file in C using the fscanf() and fgets() and fgetc() functions. All are used to read contents of a file.

Reading a character from/to a file: The data which is written into a file can also be read form the same file. fgetc() is used to read a single character from a file.

Syntax: **ch=fgetc(file_pointer);**

- ch is a character variable and fp is file pointer.

Reading a string from/to a file: The data which is written into a file in the form of strings, can also be read . fgets() is used for reading a string from the file

Syntax: fgets(s ,n,fp);

Where s is string , n is the maximum length of input string and fp is the file pointer.

Reading a data from file: For reading **formatted** data from a file, we use fscanf() function.This function is used to read any type of variable i.e. char,int,float,char,string etc.

fscanf () is formatted input function

Syntax : fscanf(fp,"format string",listofvariables);

Example: fscanf(fp,"%s%s",name,add);

- fp is file pointer.

Reading integer data from file: we use getw() function to read integer value. This is unformatted input function.

Syntax: getw(fp);

- fp is file pointer.

3.Writing a File

Before writing to a file we must have to open it in write mode.In C, when you write to a file, newline characters '\n' must be explicitly added.

Writing a character into a file: putc() is used to write a single character in the data a file.It is an ouput function.

Syntax: **fputc(ch, file_pointer)**:

- ch is a character variable and fp is file pointer.

Writing a string into a file : We can write data into a file in the form of strings. fputs() is used to writes a string to the file pointed to by file pointer.

Syntax: fputs(s,fp);

Where s is string constant and variable and fp is the file pointer.

Writing a data from file: For Writing **formatted** data to a file, we use fprintf () function. This function is used to write any type of variable i.e. char, int, float, char, string etc.

fprintf () is formatted output function.

Syntax : fprintf (fp,"format string",listofvariables);

Example: fprintf (fp,"%s%s",name,add);

- fp is file pointer.

Writing integer data to file: we use putw() function to writing data in form of integers. This is unformatted output function.

Syntax: putw (n,fp);

Where n is integer variable a nd fp is a file pointer.

4.Closing a file

When you have finished your work of reading data from or writing data onto a file, the file must be closed.

The fclose() function is used to close a file. The file must be closed after performing all the operations on it.

The general form of closing a file is:-

fclose(fp);

Where fp is a file pointer associated with a file which is to be closed.

- **End Of File (EOF)**

EOF stands for End of File. The function fgetc() returns EOF, on success.

If we want to read a complete file that is read till the data is there in the file, then end of file can be checked by using following method.

```
while(a=fgetc(fp)!-EOF

{

ch=fgetc(fp);

printf("%c",ch);

}
```