WHAT IS FILE?

File is a collection of bytes that is stored on secondary storage devices like disk. There are two kinds of files in a system. They are,

1. Text files (ASCII)
2. Binary files

- Text files contain ASCII codes of digits, alphabetic and symbols.
- Binary file contains collection of bytes (0's and 1's). Binary files are compiled version of text files.

BASIC FILE OPERATIONS IN C PROGRAMMING:

There are 4 basic operations that can be performed on any files in C programming language. They are,

1. Opening/Creating a file
2. Closing a file
3. Reading a file
4. Writing in a file

Let us see the syntax for each of the above operations in a table:

File operation	Declaration & Description
fopen() – To open a file	Declaration: FILE ***fopen** (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=**fopen** ("filename", "'mode"); Where, fp – file pointer to the data type "FILE". filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.

fclose() – To close a file	Declaration: int **fclose**(FILE *fp);
	
	fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.
	fclose (fp);

fgets() – To read a file	Declaration: char ***fgets**(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. **fgets** (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer

fprintf() – To write into a file	Declaration: int **fprintf**(FILE *fp, const char *format, …);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, “some data”); or fprintf (fp, “text %d”, variable_name);

MODE OF OPERATIONS PERFORMED ON A FILE IN C LANGUAGE:

There are many modes in opening a file. Based on the mode of file, it can be opened for reading or writing or appending the texts. They are listed below.

- r – Opens a file in read mode and sets pointer to the first character in the file. It returns null if file does not exist.
- w – Opens a file in write mode. It returns null if file could not be opened. If file exists, data are overwritten.
- a – Opens a file in append mode. It returns null if file couldn’t be opened.
- r+ – Opens a file for read and write mode and sets pointer to the first character in the file.
- w+ – opens a file for read and write mode and sets pointer to the first character in the file.
- a+ – Opens a file for read and write mode and sets pointer to the first character in the file. But, it can’t modify existing contents.

1. EXAMPLE PROGRAM FOR FILE OPEN, FILE WRITE AND FILE CLOSE IN C LANGUAGE:

```c
/ * Open, write and close a file : */
# include <stdio.h>
# include <string.h>

int main( )
{
    FILE *fp ;
    char data[50];
    // opening an existing file
    printf( "Opening the file test.c in write mode" ) ;
    fp = fopen("test.c", "w") ;
    if ( fp == NULL )
    {
        printf( "Could not open file test.c" ) ;
        return 1;
    }
    printf( "\n Enter some text from keyboard” \
             “ to write in the file test.c" ) ;
    // getting input from user
    while ( strlen ( gets( data ) ) > 0 )
    {
        // writing in the file
        fputs(data, fp) ;
        fputs("\n", fp) ;
    }
    // closing the file
    printf("Closing the file test.c") ;
    fclose(fp) ;
    return 0;
}
```

OUTPUT:

```
Opening the file test.c in write mode
Enter some text from keyboard to write in the file test.c
Hai, How are you?
Closing the file test.c
```

2. EXAMPLE PROGRAM FOR FILE OPEN, FILE READ AND FILE CLOSE IN C LANGUAGE:

This file handling C program illustrates how to read the contents of a file. Assume that, a file called “test.c” contains the following data “Hai, How are you?”. Let’s read this data using following C program.

```c
/* Open, Read and close a file: reading string by string */

# include <stdio.h>
int main( )
{
         FILE *fp ;
         char data[50] ;
         printf( "Opening the file test.c in read mode" ) ;
         fp = fopen( "test.c", "r" ) ;
         if ( fp == NULL )
         {
                 printf( "Could not open file test.c" ) ;
                 return 1;
         }
         printf( "Reading the file test.c" ) ;
         while( fgets ( data, 50, fp ) != NULL )
         printf( "%s" , data ) ;
         printf("Closing the file test.c") ;
         fclose(fp) ;
         return 0;
}
```

OUTPUT:

```
Opening the file test.c in read mode
Reading the file test.c
Hai, How are you?
Closing the file test.c
```

INBUILT FUNCTIONS FOR FILE HANDLING IN C LANGUAGE:

C programming language offers many inbuilt functions for handling files. They are given below. Please click on each function name below to know more details, example programs, output for the respective file handling function.

File handling functions	Description
fopen ()	fopen () function creates a new file or opens an existing file.
fclose ()	fclose () function closes an opened file.
getw ()	getw () function reads an integer from file.
putw ()	putw () functions writes an integer to file.
fgetc ()	fgetc () function reads a character from file.
fputc ()	fputc () functions write a character to file.
gets ()	gets () function reads line from keyboard.
puts ()	puts () function writes line to o/p screen.
fgets ()	fgets () function reads string from a file, one line at a time.
fputs ()	fputs () function writes string to a file.

feof ()	feof () function finds end of file.
fgetchar ()	fgetchar () function reads a character from keyboard.
fprintf ()	fprintf () function writes formatted data to a file.
fscanf ()	fscanf () function reads formatted data from a file.
fputchar ()	fputchar () function writes a character onto the output screen from keyboard input.
fseek ()	fseek () function moves file pointer position to given location.
SEEK_SET	SEEK_SET moves file pointer position to the beginning of the file.
SEEK_CUR	SEEK_CUR moves file pointer position to given location.
SEEK_END	SEEK_END moves file pointer position to the end of file.
ftell ()	ftell () function gives current position of file pointer.
rewind ()	rewind () function moves file pointer position to the beginning of the file.
getc ()	getc () function reads character from file.
getch ()	getch () function reads character from keyboard.
getche ()	getche () function reads character from keyboard and echoes to o/p screen.
getchar ()	getchar () function reads character from keyboard.
putc ()	putc () function writes a character to file.
putchar ()	putchar () function writes a character to screen.
printf ()	printf () function writes formatted data to screen.
sprinf ()	sprinf () function writes formatted output to string.
scanf ()	scanf () function reads formatted data from keyboard.
sscanf ()	sscanf () function Reads formatted input from a string.
remove ()	remove () function deletes a file.
fflush ()	fflush () function flushes a file.