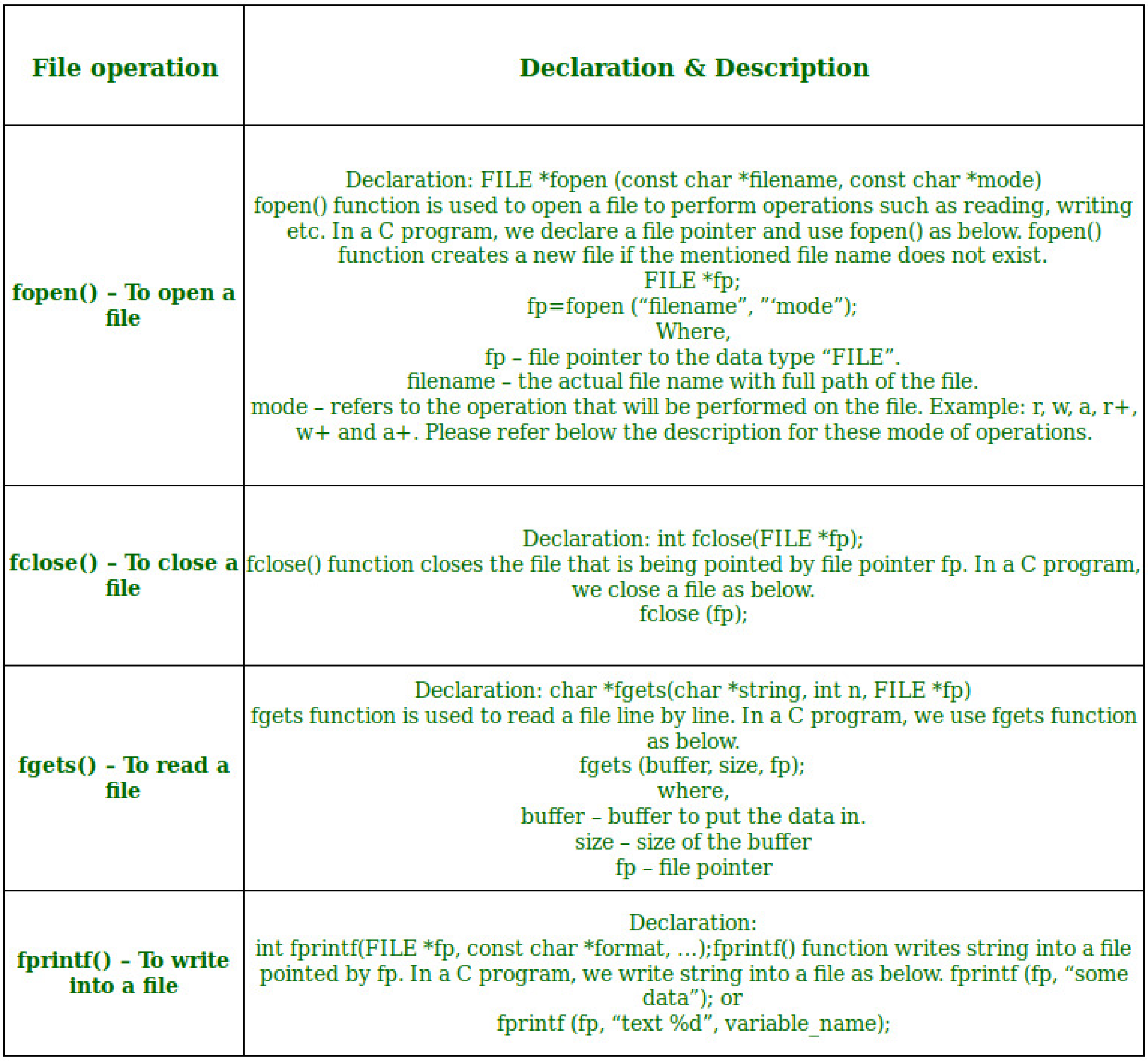
Basics of File Handling in C

So far the operations using C program are done on a prompt / terminal which is not stored anywhere. But in the software industry, most of the programs are written to store the information fetched from the program. One such way is to store the fetched information in a file. Different operations that can be performed on a file are:

1. Creation of a new file (**fopen with attributes as “a” or “a+” or “w” or “w++”)**
2. Opening an existing file (**fopen**)
3. Reading from file (**fscanf or fgetc**)
4. Writing to a file (**fprintf or fputs**)
5. Moving to a specific location in a file (**fseek, rewind**)
6. Closing a file (**fclose**)

The text in the brackets denotes the functions used for performing those operations.

Functions in File Operations:



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);

Opening or creating file

For opening a file, fopen function is used with the required access modes. Some of the commonly used file access modes are mentioned below.

File opening modes in C:

- **“r” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. If the file cannot be opened fopen( ) returns NULL.
- **“w” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

- **“a” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer that points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.
- **“r+” –** Searches file. If is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it. Returns NULL, if unable to open the file.
- **“w+” –** Searches file. If the file exists, its contents are overwritten. If the file doesn’t exist a new file is created. Returns NULL, if unable to open file.
- **“a+” –** Searches file. If the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the last character in it. If the file doesn’t exist, a new file is created. Returns NULL, if unable to open file.

As given above, if you want to perform operations on a binary file, then you have to append ‘b’ at the last. For example, instead of “w”, you have to use “wb”, instead of “a+” you have to use “a+b”. For performing the operations on the file, a special pointer called File pointer is used which is declared as

```
FILE *filePointer;
So, the file can be opened as
filePointer = fopen(“fileName.txt”, “w”)
```

The second parameter can be changed to contain all the attributes listed in the above table.

- **Reading from a file –**
  The file read operations can be performed using functions fscanf or fgets. Both the functions performed the same operations as that of printf and gets but with an additional parameter, the file pointer. So, it depends on you if you want to read the file line by line or character by character.
  And the code snippet for reading a file is as:

  ```
  FILE * filePointer;
  filePointer = fopen(“fileName.txt”, “r”);
  fscanf(filePointer, "%s %s %s %d", str1, str2, str3, &year);
  ```

- **Writing a file –**:
  The file write operations can be perfomed by the functions fprintf and fputs with similarities to read operations. The snippet for writing to a file is as :

  ```
  FILE *filePointer ;
  filePointer = fopen(“fileName.txt”, “w”);
  fprintf(filePointer, "%s %s %s %d", "We", "are", "in", 2012);
  ```

- **Closing a file –**:
  After every successful fie operations, you must always close a file. For closing a file, you have to use fclose function. The snippet for closing a file is given as :

  ```
  FILE *filePointer ;
  filePointer= fopen(“fileName.txt”, “w”);
  ```

```
---------- Some file Operations -------
fclose(filePointer)
```

Example 1: Program to Open a File, Write in it, And Close the File

```
// C program to Open a File,
// Write in it, And Close the File

# include <stdio.h>
# include <string.h>

int main( )
{

    // Declare the file pointer
    FILE *filePointer ;

    // Get the data to be written in file
    char dataToBeWritten[50]
        = "GeeksforGeeks-A Computer Science Portal for Geeks";

    // Open the existing file GfgTest.c using fopen()
    // in write mode using "w" attribute
    filePointer = fopen("GfgTest.c", "w") ;

    // Check if this filePointer is null
    // which maybe if the file does not exist
    if ( filePointer == NULL )
    {
        printf( "GfgTest.c file failed to open." ) ;
    }
    else
    {

        printf("The file is now opened.\n") ;

        // Write the dataToBeWritten into the file
        if ( strlen (  dataToBeWritten  ) > 0 )
        {

            // writing in the file using fputs()
            fputs(dataToBeWritten, filePointer) ;
            fputs("\n", filePointer) ;
        }

        // Closing the file using fclose()
        fclose(filePointer) ;

        printf("Data successfully written in file GfgTest.c\n");
        printf("The file is now closed.") ;
    }
    return 0;
}
```

Example 2: Program to Open a File, Read from it, And Close the File

```
// C program to Open a File,
// Read from it, And Close the File

# include <stdio.h>
# include <string.h>

int main( )
{

    // Declare the file pointer
    FILE *filePointer ;

    // Declare the variable for the data to be read from file
    char dataToBeRead[50];

    // Open the existing file GfgTest.c using fopen()
    // in read mode using "r" attribute
    filePointer = fopen("GfgTest.c", "r") ;

    // Check if this filePointer is null
    // which maybe if the file does not exist
    if ( filePointer == NULL )
    {
        printf( "GfgTest.c file failed to open." ) ;
    }
    else
    {

        printf("The file is now opened.\n") ;

        // Read the dataToBeRead from the file
        // using fgets() method
        while( fgets ( dataToBeRead, 50, filePointer ) != NULL )
        {

            // Print the dataToBeRead
            printf( "%s" , dataToBeRead ) ;
         }

        // Closing the file using fclose()
        fclose(filePointer) ;

        printf("Data successfully read from file GfgTest.c\n");
        printf("The file is now closed.") ;
    }
    return 0;
}
```

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

C | File Handling | Question 4

C | File Handling | Question 3

C | File Handling | Question 2

C | File Handling | Question 1

C | File Handling | Question 5

Four File Handling Hacks which every C/C++ Programmer should know

C program to copy contents of one file to another file

C | Pointer Basics | Question 13

C | Pointer Basics | Question 12

C | Pointer Basics | Question 11

C | Pointer Basics | Question 10

C | Pointer Basics | Question 17

C | Pointer Basics | Question 8

C | Pointer Basics | Question 6

C | Pointer Basics | Question 7

Improved By : RishabhPrabhu, RajanKumar3, istwasanders

Article Tags : C C Programs

Practice Tags : C

👍 13

2.7

□ To-do □ Done

Based on **17** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org