**FILE PROCESSING**

A file is a place on the disk where a group of related data is stored.Using files data can be stored permanently.

.A file named has two parts –the name of the file and the extension which is an optional part,

**Example**-**image.jpg, document.doc, abc.txt**

Thus ,a file is a container in computer storage devices used for storing data

**Q.Describe the method for declaring and opening a file**

Before using a file we have to first declare and open a file. When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and the program

**Syntax:**

**File \*fp;** //declaring a file pointer fp

**fp=fopen(“filename”,”mode”);**

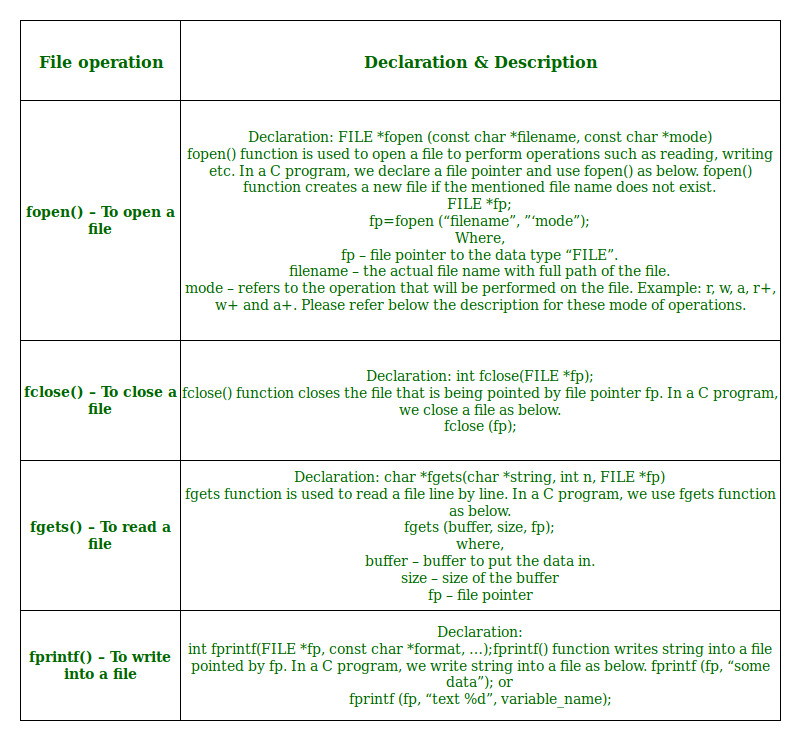
here,file is a structure,fp is a file pointer and it contains all the informationof the file to be opened,mode will be the purpose of opening a file.

**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 fgets**)
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:**



**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.

| Opening Modes in Standard I/O | | |
| --- | --- | --- |
| Mode | Meaning of Mode | During Inexistence of file |
| r | Open for reading. | If the file does not exist, fopen() returns NULL. |
| rb | Open for reading in binary mode. | If the file does not exist, fopen() returns NULL. |
| w | Open for writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| wb | Open for writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a | Open for append. Data is added to the end of the file. | If the file does not exist, it will be created. |
| ab | Open for append in binary mode. Data is added to the end of the file. | If the file does not exist, it will be created. |
| r+ | Open for both reading and writing. | If the file does not exist, fopen() returns NULL. |
| rb+ | Open for both reading and writing in binary mode. | If the file does not exist, fopen() returns NULL. |
| w+ | Open for both reading and writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| wb+ | Open for both reading and writing in binary mode. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a+ | Open for both reading and appending. | If the file does not exist, it will be created. |
| ab+ | Open for both reading and appending in binary mode. | If the file does not exist, it will be created. |

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 scanf 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)

**Reading and writing to a text file**

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that fprint() and fscanf() expects a pointer to the structure FILE.

**Example 1: Write to a text file**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num;

FILE \*fptr;

// use appropriate location if you are using MacOS or Linux

fptr = fopen("C:\\program.txt","w");

if(fptr == NULL)

{

printf("Error!");

exit(1);

}

printf("Enter num: ");

scanf("%d",&num);

fprintf(fptr,"%d",num);

fclose(fptr);

return 0;

}

This program takes a number from the user and stores in the file program.txt.

After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

**Example 2: Read from a text file**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num;

FILE \*fptr;

if ((fptr = fopen("C:\\program.txt","r")) == NULL){

printf("Error! opening file");

// Program exits if the file pointer returns NULL.

exit(1);

}

fscanf(fptr,"%d", &num);

printf("Value of n=%d", num);

fclose(fptr);

return 0;

}

This program reads the integer present in the program.txt file and prints it onto the screen.

If you successfully created the file from **Example 1**, running this program will get you the integer you entered.

Other functions like fgetchar(), fputc() etc. can be used in a similar way.

**Example 3:** **Program to Open a File, Write in it, And Close the File**

|  |
| --- |
| # include <stdio.h>  # include <string.h>   int main( )  {    FILE \*filePointer ; // **Declare the file pointer**    // **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 4:** **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;  } |

**Example5:The following program will create a text file using the putc()function**

#include<stdio.h>

int main()

{

char ch;

FILE \*fp;

fp=fopen("jncollege.txt","w");

printf("\n type the text and press enter key at the end");

while((ch=getchar())!='\n')

{

putc(ch,fp);

}

fclose(fp);

**Output:**

**Type the text and press enter key at the end.**

**JNcollegepgdca**

**If we want to read the text file “JNcollege.txt” and count the number of vowels,then the program can be written as follows:**

#include<stdio.h>

int main()

{

char ch;

int count=0;

FILE \*fp;

fp=fopen("jncollege.txt","r");

printf("\n the content ofthe text file is as follows:\n\n\n");

while(!feof(fp)) //while reading data,feof()locate the end of file

{

ch=getc(fp);

printf("%c",ch);

switch(tolower(ch))

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

count++;

break;

}

}

printf("\n\n\n the number of vowels counted in the text file =%d",count);

fclose(fp);

}

**Output:**

**The content of the text file is as follows:**

**JNcollegepgdca**

**The number of vowels counted in the text file=4**

**Reading and writing to a binary file**

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.

**Writing to a binary file**

To write into a binary file, you need to use the fwrite() function. The functions take four arguments:

1. address of data to be written in the disk
2. size of data to be written in the disk
3. number of such type of data
4. pointer to the file where you want to write.

fwrite(addressData, sizeData, numbersData, pointerToFile);

**Reading from a binary file**

Function fread() also take 4 arguments similar to the fwrite() function as above.

fread(addressData, sizeData, numbersData, pointerToFile);

**Why files are needed?**

* When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
* If you have to enter a large number of data, it will take a lot of time to enter them all.  
  However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.
* You can easily move your data from one computer to another without any changes.

**Types of Files**

When dealing with files, there are two types of files you should know about:

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 you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

**2. Binary files**

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

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

They can hold a higher amount of data, are not readable easily, and provides better security than text files.