fseek() vs rewind() in C

In C, fseek() should be preferred over rewind().

The rewind function sets the file position indicator for the stream pointed to by stream to the beginning of the file. It is equivalent to

(void)fseek(stream, 0L, SEEK_SET)

EOF, getc() and feof() in C

In C/C++, getc() returns EOF when end of file is reached. getc() also returns EOF when it fails. So, only comparing the value returned by getc() with EOF is not sufficient to check for actual end of file. To solve this problem, C provides feof() which returns non-zero value only if end of file has reached, otherwise it returns 0.
For example, consider the following C program to print contents of file test.txt on screen. In the program, returned value of getc() is compared with EOF first, then there is another check using feof(). By putting this check, we make sure that the program prints “End of file reached” only if end of file is reached. And if getc() returns EOF due to any other reason, then the program prints “Something went wrong”

fopen() for an existing file in write mode

In C, fopen() is used to open a file in different modes. To open a file in write mode, “w” is specified. When mode “w” is specified, it creates an empty file for output operations.

What if the file already exists?

If a file with the same name already exists, its contents are discarded and the file is treated as a new empty file. For example, in the following program, if “test.txt” already exists, its contents are removed and “GeeksforGeeks” is written to it.

ftell() in C with example

ftell() in C is used to find out the position of file pointer in the file with respect to starting of the file. Syntax of ftell() is:

long ftell(FILE *pointer)

Consider below C program. The file taken in the example contains the following data :
“Someone over there is calling you. We are going for work. Take care of yourself.” (without the quotes)
When the fscanf statement is executed word “Someone” is stored in string and the pointer is moved beyond “Someone”. Therefore ftell(fp) returns 7 as length of “someone” is 6.

Read/Write structure to a file in C

For writing in file, it is easy to write string or int to file using **fprintf** and **putc**, but you might have faced difficulty when writing contents of struct. **fwrite** and **fread** make task easier when you want to write and read blocks of data.

1. **fwrite :** Following is the declaration of fwrite function

   size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
   ptr - This is pointer to array of elements to be written
   size - This is the size in bytes of each element to be written
   nmemb - This is the number of elements, each one with a size of size bytes
   stream - This is the pointer to a FILE object that specifies an

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 (**filePointerrintf or filePointeruts**)
5. Moving to a specific location in a file (**fseek, rewind**)
6. Closing a file (**fclose**)

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. 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.

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);

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 filePointerrintf and filePointeruts with similarities to read operations. The snippet for writing to a file is as :

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

What is data type of FILE in C ?

In C language, while file handling is done a word FILE is used. What is FILE?
FILE *fp1, *fp2;

While doing file handling we often use FILE for declaring the pointer in order to point to the file we want to read from or to write on. As we are declaring the pointer of type FILE so we can say it is data type, but what kind of data type? A FILE is a type of structure typedef as FILE. It is considered as **opaque data type** as its implementation is hidden. We don't know what constitutes the type, we only use pointer to the type and library knows the internal of the type and can use the data.

fseek() in C/C++ with example

fseek() is used to move file pointer associated with a given file to a specific position.

int fseek(FILE *pointer, long int offset, int position)
pointer: pointer to a FILE object that identifies the stream.
offset: number of bytes to offset from position
position: position from where offset is added.

returns:
zero if successful, or else it returns a non-zero value

position defines the point with respect to which the file pointer needs to be moved. It has three values:
SEEK_END : It denotes end of the file.
SEEK_SET : It denotes starting of the file.
SEEK_CUR : It denotes file pointer's current position.

fgetc() and fputc() in C

fgetc()

fgetc() is used to obtain input from a file single character at a time. This function returns the number of characters read by the function. It returns the character present at position indicated by file pointer. After reading the character, the file pointer is advanced to next character. If pointer is at end of file or if an error occurs EOF file is returned by this function.

Syntax:

int fgetc(FILE *pointer)

pointer: pointer to a FILE object that identifies
the stream on which the operation is to be performed.

Using fputc()

fputc() is used to write a single character at a time to a given file. It writes the given character at the position denoted by the file pointer and then advances the file pointer.
This function returns the character that is written in case of successful write operation else in case of error EOF is returned.
Syntax:

int fputc(int char, FILE *pointer)

char: character to be written.
This is passed as its int promotion.
pointer: pointer to a FILE object that identifies the
stream where the character is to be written.

tmpfile() function in C

In C Programming Language, the tmpfile() function is used to produce/create a temporary file.

- tmpfile() function is defined in the “stdio.h” header file.
- The created temporary file will automatically be deleted after the termination of program.
- It opens file in binary update mode i.e., wb+ mode.
- The syntax of tmpfile() function is:

  FILE *tmpfile(void)

- The tmpfile() function always returns a pointer after the creation of file to the temporary file. If by chance temporary file can not be created, then the tmpfile() function returns NULL pointer.

rename function in C/C++

rename() function is used to change the name of the file or directory i.e. from **old_name** to **new_name** without changing the content present in the file. This function takes name of the file as its argument.
If **new_name** is the name of an existing file in the same folder then the function may either fail or override the existing file, depending on the specific system and library implementation.

Syntax:

int rename (const char *old_name, const char *new_name);

Parameters:
old_name : Name of an existing file to be renamed.
new_name : String containing new name of the file.

Return:
Return type of function is an integer. If the file is renamed successfully, zero is returned. On failure, a nonzero value is returned.

Assume that we have a text file having name **geeks.txt**, having some content. So, we are going to rename this file, using the below C program present in the same folder where this file is present.

fsetpos() (Set File Position) in C

The **fsetpos()** function moves the file position indicator to the location specified by the object pointed to by position. When **fsetpos()** is executed ,the end-of-file indecator is reset.

Declaration

int fsetpos(FILE *stream, const fpos_t *position)

Parameters –

- **stream –** This is the pointer to a FILE object that identifies the stream.
- **position –** This is the pointer to a fpos_t object containing a position previously obtained with fgetpos.

Return – If it successful, it return **zero** otherwise returns nonzero value.

fgets() and gets() in C language

For reading a string value with spaces, we can use either gets() or fgets() in C programming language. Here, we will see what is the difference between gets() and fgets().

fgets()

It reads a line from the specified stream and stores it into the string pointed to by str. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.

Syntax :

char *fgets(char *str, int n, FILE *stream)
str : Pointer to an array of chars where the string read is copied.
n : Maximum number of characters to be copied into str
(including the terminating null-character).
***stream :** Pointer to a FILE object that identifies an input stream.
stdin can be used as argument to read from the standard input.

returns : the function returns str

It follow some parameter such as Maximum length, buffer, input device reference.

- It is **safe** to use because it checks the array bound.
- It keep on reading until new line character encountered or maximum limit of character array.

Example : Let’s say the maximum number of characters are 15 and input length is greater than 15 but still fgets() will read only 15 character and print it.

```
// C program to illustrate
// fgets()
#include <stdio.h>
```

```
#define MAX 15
int main() {
 char buf[MAX];
    fgets(buf, MAX, stdin);
    printf("string is: %s\n", buf);
    return 0;
}
```

Since fgets() reads input from user, we need to provide input during runtime.

Input:
Hello and welcome to GeeksforGeeks

Output:
Hello and welc

gets()

Reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached.

Syntax:

char * gets (char * str);
str : Pointer to a block of memory (array of char)
where the string read is copied as a C string.
returns : the function returns str

- It is not safe to use because it does not check the array bound.
- It is used to read string from user until newline character not encountered.

Example : Suppose we have a character array of 15 characters and input is greater than 15 characters, gets() will read all these characters and store them into variable.Since, gets() do not check the maximum limit of input characters, so at any time compiler may return buffer overflow error.

```
// C program to illustrate
// gets()
#include <stdio.h>
#define MAX 15

int main()
{
    char buf[MAX];

    printf("Enter a string: ");
    gets(buf);
    printf("string is: %s\n", buf);

    return 0;
}
```

Since gets() reads input from user, we need to provide input during runtime.

Input:
Hello and welcome to GeeksforGeeks

Output:
Hello and welcome to GeeksforGeeks