File I/O in C

fopen() function is used to open a file in writing, reading or appending mode, before performing operation on it.

Syntax :-

```
File *ptr;
ptr = fopen("file name", "mode");
```

mode	work
"r"	used to open a previously created file in reading only mode.
"w"	used to open a file previously created file in writing only mode or create a new file and open it in writing mode.
"a"	used to open a file previously created file in appending only mode or create a new file and open it in appending mode.

"r+"	used for reading, writing new content and modifying existing content on a file
"w+"	Used for writing new content, reading them back and modifying existing contents of the file.
"a+"	reading existing contents and appending at end of line, can't modify existing content.

If we open a file using fopen(), after it use it work complete we must close it using fclose().

Syntax:-

fclose(ptr);

Till now we have learnt how to open a file in specific mode and based on those modes we can work with files like writing to files, reading from files, appending to files.

We use various functions to do so, and we will learn about them one by one in table.

NOTE :- When we first open a file we get pointer to the first character of the file, and as we work on the file position of this pointer changes automatically.

If we read a character or string, it will read from the position where pointer is pointing and after that it moves to next character or string so, we can read all chacter or string in a file using loop by running the loop until we reach end of file (at end of file we get EOF character if we are reading character or NULL if reading string).

Also if we have a pointer to file pointing at first character of the file and we write to that file or append then pointer automatically starts to point at the end of file as writing or appending starts from there. And after that if we want to read from that file we can't do that directly as pointer is at end of file.

There are two options -

First read then write and after writing don't read.

Second use fseek() function or rewind() function to change the position of pointer then read.

→ Function used to write to/append to file.

→ To write one character :-

• fputc() :-

Used to write/append one character at a time in file.

Syntax -

fputc(character, file pointer);

→ To write whole string :-

• fputs() :-

Used to write/append one string at a time in file.

Syntax -

fputs(string, file pointer);

To put new file we have to again use fputs() as it can write only one string at a time.

fputs("\n", file pointer);

→ To write formmated statement!-

• fprintf() :-

This function works like printf() but to write in file not on output terminal/consol.

It can used to write multiple string, integer or other type data at once in file.

Syntax!-

fprintf(file pointer, formate specifier, list);

In formate specifier we can use all formate specifier which can be used in printf().

ex-

printf(ptr, "%s \n %d", arr, n);

→ Functions to read from a file :-

→ To read one character :-

• fgetc() :-

Used to read one character at a time from file.

```
char ch;
ch = fgetc(file pointer);
```

It will return charcter pointed by file pointer at that point.
We use ~~for~~ loop to print whole file character by character until we get EOF character.

```
do{
    ch = fgetc(ptr);
    printf("%c", ch);
}while(ch != EOF);
```

Instead of using EOF character to find end of EOF, we can use feof() function, this function return 'True' if we read end of file.

```
while (feof(Ptr) == 0)
{
    ch = fgetc(Ptr);
    printf("%c", ch);
}
```

→ To read one string:-

- fgets():-

Used to read one string at a time.

```
fgets(string, min_size_string, file pointer);
```

Char array in which string will be stored

min size of string to be read.

We can run loop to read more than one / all strings in the file.

```
while(fgets(String, 50, Ptr) != NULL)
{
    printf("%s", String);
}
```

Or we can use feof() function

```
while (feof(Ptr) == 0)
{
    fgets(string, 50, Ptr);
    fprintf("%s\n", string);
}
```

In this there is a logical error, last string is printed 2 times.

→ To read formatted statement :-

. fscanf() :-

Used to read integer, float, character and other type of data from file at once.

Syntax :-

fscanf(file pointer, "format speci.", list of variables);

We don't use fscanf() to read multi word string from file as, like scanf() it stops reading string if incounter space, tab or new line. So we use fgets() for this.

* WAP to write multiple string in file.

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

int main()
{
    FILE *ptr;
    char String[50];
    ptr=fopen("Text.txt","at");

    if(NULL==ptr)
    {
        Printf("File can't be opened\n");
    }
```

```
printf("Write strings to write \n");
printf("To stop enter two times \n");

while(strlen(gets(string)) > 0)
{
    fputs(string, ptr);
}

fclose(ptr);

return 0;
}
```

* WAP to read multiple string from file

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

int main()
{
    FILE * ptr;
    char string[100];
    ptr = fopen("Text.txt", "a+");
```

```
if(NULL == ptr)
{
    printf("File can't be opened\n");
}  exit(1);

printf("content of the file :- \n");

while( fgets(string, 100, ptr) != NULL)
{
    printf("%.s\n", string);
}

fclose(ptr);

}
```

- Rewind():-

It changes the position of file pointer from elsewhere to the beginning of file.

Q. WAP to write/append in the file and then read all string of the file.

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

int main()
{
   FILE *ptr;
   char string[100];
   ptr=fopen("Text.txt","a+");

   if(NULL==ptr)
   {
      printf("File can't be opened \n");
      exit(1);
   }
   printf("input your string and to
         stop press enter two times \n");

   while(strlen(gets(string))>0)
   {
      fputs(string,ptr);
      fputs("\n",ptr);
   }
   rewind();
```

```
printf(" contents of this files are \n");
while(fgets(string,100,ptr) != NULL)
{
    printf("%s \n", string);
}
fclose(ptr);
return 0;
}
```

Q. WAP to take input in structure, print it's data in the file then read all it's data.

→
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

int main()
{
    FILE *ptr;
    char ch;
    typedef struct employee
    {
        char name[30];
        int Id[10];
        float salary;
    } employee;
```

```
employee e;
Ptr = fopen("employee.txt", "a+");

if(NULL == Ptr)
{
  Printf("File can't be opened");
  exit(1);
}

do
{
  Printf("Input name of character\n");
  fflush(stdin);
  gets(e.name);
// converting space in name in (_)underscore
// to read it using scanf.
  for(int i=0; i<strlen(e.name); i++)
  {
    if(e.name[i]==' ')
      e.name[i]='_';
  }
Printf("Input id and salary of employee\n");
Scanf("%d%f", e.Id, &e.salary);
fprintf(Ptr, "%s %d %f\n", e.name, e.Id,
                          e.salary);
```

```
printf("input 'y' if you want to
        enter another set of data or
        other character if not \n");
fflush(stdin);
scanf ("%c",&ch);
} while(ch=='y' || ch=='Y');

rewind (ptr);
while(fscanf("%s%d%f", e.name, e.Id,
                    &e.salary)!=EO
{
    for(int i=0; i<strlen(e.name); i++)
    {
        if(e.name[i]=='_')
        {
            e.name[i]=' ';
        }
    }
printf("%s %d %.2f\n", e.name,
                e.Id, e.salary);
}
fclose(ptr);
return 0;
}
```

There is another method to print contents of file in this case, using fgets().

```
char string[100];
while (fgets(string, 500, ptr) != NULL)
{
    printf("%s\n", string);
}
```

In this case we also don't need to change space in the name to '_' underscore before writing in the file.

How to read content of whole file.

We know, when we use f we can read content of a file using functions like fscanf(), fgets(), fgetc() etc. but when we do so, we only get only first string or first character of the file. How can we get content of whole string.

We will do so learn how to do so using.

fgetc() :-

This function reads character pointed by the function pointer at that time. and then pointer moves to next character. We can read whole content of the file, using this function by reading character one by one. We will use one loop which read character until it reaches end of file. When it reaches end it return EOF character so, that we can know that we reached the end.

Or we can use feof() function which return true if we reached end.

using EOF

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    char ch;
    FILE *ptr;
    ptr = fopen("Text.txt","r");

    if(NULL==ptr)
    {
        printf("File can't be opened\n");
        exit(1);
    }

    do
    {
    ch = fgetc(ptr);
    printf("%c",ch);
    } while(ch!=EOF);
    fclose(ptr);
    return 0;
}
```

Using feof() fn

fgets():-

This function reads one string at a time from the file.

Syntax:-

fgets(string, min_size_string, file pointer);

To read whole content of the file if we run loop until we reach end of file. At end it return NULL character.

Code is written previously.

fscanf():-