File Handling in C

File Handling in c language is *used to open, read, write, search or close file*. It is used for permanent storage.

Advantage of File

It *will contain the data even after program exit*. Normally we use variable or array to store data, but data is lost after program exit. Variables and arrays are non-permanent storage medium whereas file is permanent storage medium.

Functions for file handling

There are many functions in C library to open, read, write, search and close file. A list of file functions are given below:

No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into file
3	fscanf()	reads data from file
4	fputc()	writes a character into file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position

11	rewind()	sets the file pointer to the beginning of the file

Opening File: fopen()

The fopen() function is used to open a file. The syntax of fopen() function is given below:

1. FILE *fopen(const char * filename, const char * mode);

You can use one of the following modes in the fopen() function.

Mode	Description
r	opens a text file in read mode
w	opens a text file in write mode
a	opens a text file in append mode
r+	opens a text file in read and write mode
w+	opens a text file in read and write mode
a+	opens a text file in read and write mode
rb	opens a binary file in read mode
wb	opens a binary file in write mode
ab	opens a binary file in append mode
rb+	opens a binary file in read and write mode
wb+	opens a binary file in read and write mode
ab+	opens a binary file in read and write mode

Closing File: fclose()

The fclose() function is used to close a file. The syntax of fclose() function is given below:

1. `int fclose( FILE *fp );`

C fprintf() and fscanf()

C fprintf() and fscanf() example

C fputc() and fgetc()

C fputc() and fgetc() example

C fputs() and fgets()

C fputs() and fgets() example

C fseek()

C fseek() example

C fprintf() and fscanf()

Writing File : fprintf() function

The fprintf() function is used to write set of characters into file. It sends formatted output to a stream.

Syntax:

1. `int fprintf(FILE *stream, const char *format [, argument, ...])`

Example:

```
#include <stdio.h>
main(){
```

```
  FILE *fp;
  fp = fopen("file.txt", "w");          //opening file
  fprintf(fp, "Hello file by fprintf...\n");//writing data into file
  fclose(fp);//closing file
}
```

Reading File : fscanf() function

The fscanf() function is used to read set of characters from file. It reads a word from the file and returns EOF at the end of file.

Syntax:

1. `int fscanf(FILE *stream, const char *format [, argument, ...])`

Example:

```
#include <stdio.h>
main(){
  FILE *fp;
  char buff[255];//creating char array to store data of file
  fp = fopen("file.txt", "r");
  while(fscanf(fp, "%s", buff)!=EOF){
  printf("%s ", buff );
  }
  fclose(fp);
}
```

Output:

```
Hello file by fprintf...
```

C File Example: Storing employee information

Let's see a file handling example to store employee information as entered by user from console. We are going to store id, name and salary of the employee.

```
#include <stdio.h>
void main()
```

```
{
   FILE *fptr;
   int id;
   char name[30];
   float salary;
   fptr = fopen("emp.txt", "w+");/*  open for writing */
   if (fptr == NULL)
   {
      printf("File does not exists \n");
      return;
   }
   printf("Enter the id\n");
   scanf("%d", &id);
   fprintf(fptr, "Id= %d\n", id);
   printf("Enter the name \n");
   scanf("%s", name);
   fprintf(fptr, "Name= %s\n", name);
   printf("Enter the salary\n");
   scanf("%f", &salary);
   fprintf(fptr, "Salary= %.2f\n", salary);
   fclose(fptr);
}
```

Output:

```
Enter the id
1
Enter the name
sonoo
Enter the salary
120000
```

Now open file from current directory. For windows operating system, go to TC\bin directory, you will see emp.txt file. It will have following information.

emp.txt

```
Id= 1
Name= sonoo
Salary= 120000
```

C fputc() and fgetc()

Writing File : fputc() function

The fputc() function is used to write a single character into file. It outputs a character to a stream.

Syntax:

```
int fputc(int c, FILE *stream)
```

Example:

```
#include <stdio.h>
main(){
  FILE *fp;
  fp = fopen("file1.txt", "w");//opening file
  fputc('a',fp);//writing single character into file
  fclose(fp);//closing file
}
```

file1.txt

```
a
```

Reading File : fgetc() function

The fgetc() function returns a single character from the file. It gets a character from the stream. It returns EOF at the end of file.

Syntax:

```
int fgetc(FILE *stream)
```

Example:

```
#include<stdio.h>
#include<conio.h>
void main(){
FILE *fp;
```

```
5. char c;
6. clrscr();
7. fp=fopen("myfile.txt","r");
8.
9. while((c=fgetc(fp))!=EOF){
10. printf("%c",c);
11. }
12. fclose(fp);
13. getch();
14. }
```

myfile.txt

```
this is simple text message
```

C fputs() and fgets()

The fputs() and fgets() in C programming are used to write and read string from stream. Let's see examples of writing and reading file using fgets() and fgets() functions.

Writing File : fputs() function

The fputs() function writes a line of characters into file. It outputs string to a stream.

Syntax:

```
1. int fputs(const char *s, FILE *stream)
```

Example:

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main(){
4. FILE *fp;
5. clrscr();
6.
7. fp=fopen("myfile2.txt","w");
8. fputs("hello c programming",fp);
```

```
9.
10. fclose(fp);
11. getch();
12. }
```

myfile2.txt

```
hello c programming
```

Reading File : fgets() function

The fgets() function reads a line of characters from file. It gets string from a stream.

Syntax:

```
1. char* fgets(char *s, int n, FILE *stream)
```

Example:

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main(){
4. FILE *fp;
5. char text[300];
6. clrscr();
7.
8. fp=fopen("myfile2.txt","r");
9. printf("%s",fgets(text,200,fp));
10.
11. fclose(fp);
12. getch();
13. }
```

Output:

```
hello c programming
```

C fseek() function

The fseek() function is used to set the file pointer to the specified offset. It is used to write data into file

at desired location.

Syntax:

1. `int fseek(FILE *stream, long int offset, int whence)`

There are 3 constants used in the fseek() function for whence: SEEK_SET, SEEK_CUR and SEEK_END.

Example:

```
#include <stdio.h>
void main(){
  FILE *fp;

  fp = fopen("myfile.txt","w+");
  fputs("This is javatpoint", fp);

  fseek( fp, 7, SEEK_SET );
  fputs("sonoo jaiswal", fp);
  fclose(fp);
}
```

myfile.txt

```
This is sonoo jaiswal
```

C rewind() function

The rewind() function sets the file pointer at the beginning of the stream. It is useful if you have to use stream many times.

Syntax:

1. `void rewind(FILE *stream)`

Example:

File: file.txt

1. this is a simple text

File: rewind.c

```
#include<stdio.h>
#include<conio.h>
void main(){
FILE *fp;
char c;
clrscr();
fp=fopen("file.txt","r");

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

rewind(fp);//moves the file pointer at beginning of the file

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

fclose(fp);
getch();
}
```

Output:

```
this is a simple textthis is a simple text
```

As you can see, rewind() function moves the file pointer at beginning of the file that is why "this is simple text" is printed 2 times. If you don't call rewind() function, "this is simple text" will be printed only once.

C ftell() function

The ftell() function returns the current file position of the specified stream. We can use ftell() function to get the total size of a file after moving file pointer at the end of file. We can use SEEK_END constant to move the file pointer at the end of file.

Syntax:

```
long int ftell(FILE *stream)
```

Example:

File: ftell.c

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  void main (){
4.    FILE *fp;
5.    int length;
6.    clrscr();
7.    fp = fopen("file.txt", "r");
8.    fseek(fp, 0, SEEK_END);
9.
10.   length = ftell(fp);
11.
12.   fclose(fp);
13.   printf("Size of file: %d bytes", length);
14.   getch();
15. }
```

Output:

```
Size of file: 21 bytes
```