Chapter-13
FILE HANDLING

File: A file is a container in computer storage devices used for storing data. File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program. The following operations can be performed on a file.

- Creation of the new file
- Opening an existing file
- Reading from the file
- Writing to the file
- Deleting the file

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.

Functions for file handling:

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

No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fputs()	writes a string into the file
7	fgets()	reads a string value with spaces from the file
8	fclose()	closes the file
9	remove()	Removes a file
10	rename()	Rename a file

MODES Used In File Handling:

- r : opens a text file in reading mode.
- w : opens or creates a text file in writing mode.
- a : opens a text file in append mode.
- a+ : opens a text file in both reading and appending mode. New data is appended at the end of the file and does not overwrite the existing content.

NOTE: Writing on the file will overwrite the previous content if we use "w" (write) mode and in "a" (append) mode the previous contents appended with the new content.

Working with files:
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 *fptr;
Example: FILE *fp;

Opening a file - for creation and edit:
Opening a file is performed using the fopen() function defined in the stdio.h header file.

1) **fopen():**

- The **fopen()**in C is a library function that is used to open a file to perform various operations which include reading, writing etc. along with various modes.
- If the file exists then the particular file is opened else a new file is created.

Syntax:

FILE pointer=fopen (file_name, mode);

Example:

FILE *fp;

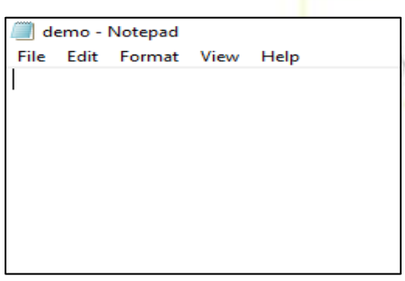
fp= fopen("demo.txt","w");

Here, We are creating a text file named "demo.txt" in write "w" mode.

Program:

```
#include<stdio.h>
int main()
{
        FILE *fp;
        fp=fopen("demo.txt","w");
        printf("file created successfully");
          return 0;
}
```

Output:
File Created Successfully.



"Demo.txt" file which we have created.

Closing a File:

The file (both text and binary) should be closed after reading/writing.
Closing a file is performed using the fclose() function.

2. fclose(): Whenever we open a file in read or write mode then we perform appropriate operations on file and when file is no longer needed then we close the file and fclose() function is used to close a file.

Syntax:
fclose(file_pointer);

Program:

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

```
        fp=fopen("demo.txt","w");
         printf("file created successfully\n");
        fclose(fp);
        printf("file closed");
        return 0;
}
```

Reading and writing to a text file:

3) fputc(): This Function is used to write a character into the file.

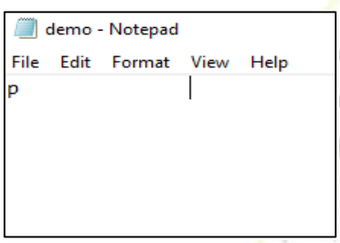
syntax:

fputc(character, file_pointer);

Program:

```
#include<stdio.h>
int main()
{
   FILE *fp;
   fp=fopen("demo.txt","w");
            fputc('p',fp); //inserting 'p' in "demo.txt" file
   printf("character inserted successfully.");
   return 0;
}
```

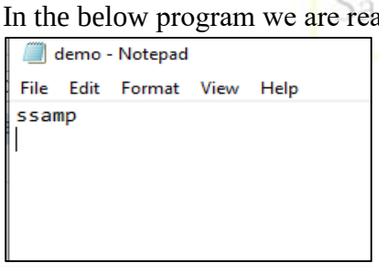
Output: character inserted successfully.



4) fgetc():

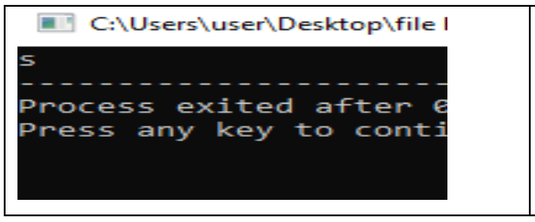
- This function is used to read characters from the file and **while reading something from the file the mode should be "r" i.e. read mode.**
- EOF represents END OF FILE and its value is -1.

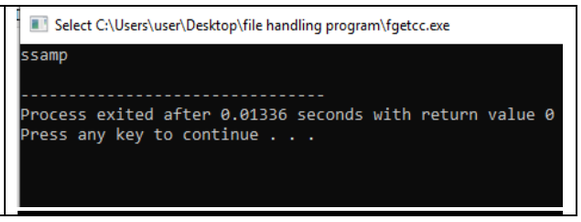
In the below program we are reading data from "demo.txt" file.



Program without EOF	**Program with EOF**

#include<stdio.h>
int main()
{
FILE *fp;
char ch;
fp=fopen("demo.txt","r");
ch = fgetc(fp);
printf("%c",ch);}
return 0;}
Note: if you are not using Any Condition using EOF then only first character will be displayed.
	#include<stdio.h>
	int main(){
	FILE *fp;
	char ch;
	fp=fopen("demo.txt","r");
	while(1){
	ch = fgetc(fp);
	if(ch == EOF)
	break;
	else
	printf("%c",ch);
	}return 0;}





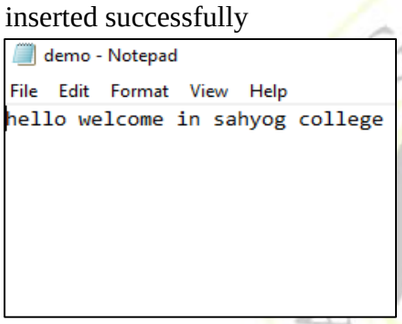
5) **fputs():** This function is used to write a string in file.

Syntax:
fputs("string",file_pointer);

Program:

```
#include<stdio.h>
int main()
{
      FILE *fp;
      fp=fopen("demo.txt","w");
       fputs("hello welcome in sahyog college",fp);
       printf("string inserted successfully");
      return 0;
}
```

Output:
string inserted successfully

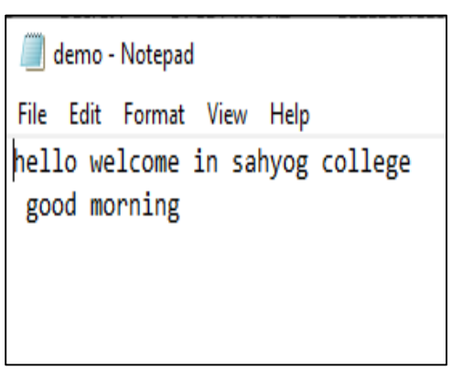


6) fgets():

- This function reads a string from the file and returns NULL at the end.

Syntax:
fgets(string_var, Size, file-pointer);

program:
program to read string from "demo.txt" file.



Program without Condition	Program with Condition

#include<stdio.h>
int main()
{
FILE *fp;
char a[200];
fp=fopen("demo.txt","r");
fgets(a,12,fp);
printf("%s",a);
return 0;
}
 	
Note: if you are not using Any Condition using NULL then only first line will be displayed.
	#include<stdio.h>
	int main()
	{
	FILE *fp;
	char a[200];
	fp=fopen("demo.txt","r");
	while(fgets(a,22,fp)!=NULL)
	{
	printf("%s",a);
	}
	return 0;
	}

C:\Users\user\Desktop\file handling program\fputss.e hello welcome in sahy ---------- Process exited after 0.009747 seconds w Press any key to continue . . .	C:\Users\user\Desktop\file handling program hello welcome in sahyog college good morning ---------- Process exited after 0.01366 seco Press any key to continue . . .

7) fprintf():

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

Syntax:

- fprintf(file_pointer, ”string”);
- fprintf(file-pointer, ”format-specifiers” ,variable-name);

Program:

PROGRAM 1:	OUTPUT

#include <stdio.h>
int 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
return 0;
}
	file - Notepad
	File Edit Format View Help
	Hello file by fprintf...

PROGRAM 2:	**OUTPUT**

#include <stdio.h>
int main(){
FILE *fp;
int a=10,b=20;
fp = fopen("file.txt", "a+");//opening file in append mode
fprintf(fp, "%d %d\n",a,b);
fclose(fp);//closing file
return 0;
}
	file - Notepad
	File Edit Format View Help
	10 20

8) fscanf():

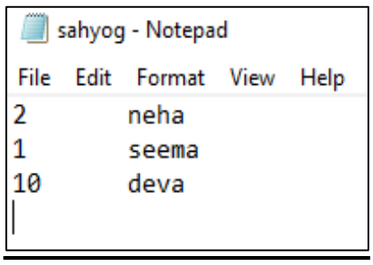
The fscanf() function is used to reads formatted input from a file and returns EOF at the end of file.

Syntax:

fscanf(file-pointer, ”format specifiers”, variable_names);

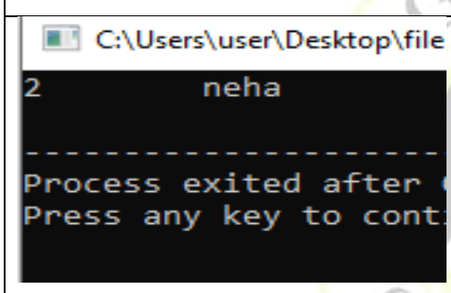
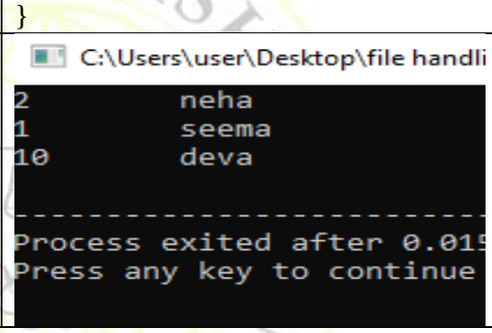
program:

program to read data from “sahyog.txt” file.

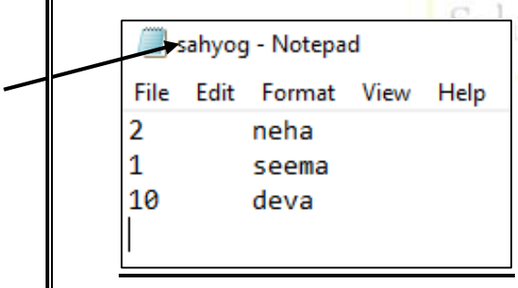


Program without Condition	Program with Condition

#include <stdio.h>
int main()
{
FILE *g;
g=fopen("sahyog.txt","r");
fscanf(g,"%d%s",&id,name);
printf("\n id is %d\n name is %s",id,name);
return 0;
}
Note: if you are not using Any Condition using NULL then only one record will be displayed.
	#include <stdio.h>
	int main()
	{
	int id;
	char name[30];
	FILE *g;
	g=fopen("sahyog.txt","r");
	while((fscanf(g,"%d%s",&id,name)!=EOF))
	{
	printf("%d\t %s\n",id,name);
	}
	return 0;
	}

9) rename(): This function is used to rename a file.



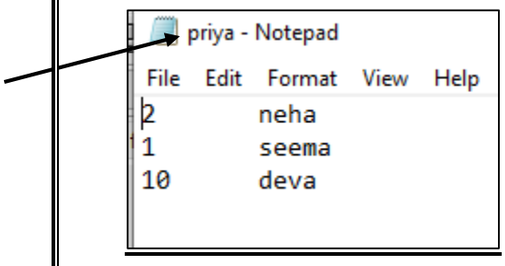
Syntax:

rename(old_name,New_name);

program:

```
#include <stdio.h>
int main()
{
rename("sahyog.txt", "priya.txt"); // renaming sahyog to priya
printf("renamed");
return 0;
}
```

Output:



10. **remove():** This function is used to remove (delete) a file.

Syntax:

remove("file name");

program:

```
#include <stdio.h>
int main()
{
remove("demo.txt");
printf(" file removed successfully.");
return 0;
}
```

Output:

File removed successfully.

***