Files

Concept of a file, Opening and Closing files, file input / output functions (standard library input / output functions for text files)

Definition:

FILE is a predefined structure data type which is defined in library file called stdio.h.

(OR)

FILE is a set of records that can be accessed through a set of library functions.

Syntax:

*FILE *filepointerobject;*

Example:

*FILE *fp;* // where FILE is a keyword and "fp" is a file pointer object

✓ C supports a number of functions that have the ability to perform basic file operations, which include:

1. Naming a file
2. Opening a file
3. Reading from a file
4. Writing data into a file
5. Closing a file

File operation functions in C:

Function Name	Operation
`fopen()`	Creates a new file for use Opens a new existing file for use *Ex: fp=fopen("gitam.txt", "w");*
`fclose()`	Closes a file which has been opened for use *Ex: fclose(fp);*
`fgetc()` or `getc()`	Reads a character from a file *Ex: ch=fgetc(fp);* *or ch=getc(fp);*
`fputc()` or `putc()`	Writes a character to a file *Ex: fputc(ch,fp);* *or putc(ch,fp);*
`fprintf()`	Writes a set of data values to a file *Ex: fprintf(fp,"%d",n);*
`fscanf()`	Reads a set of data values from a file *Ex: fscanf(fp,"%d",&n);*

`getw( )` *// Its not `fgetw()`*	Reads only integer from a file *`Ex: n=getw(fp);`*
`putw()` *// Its not `fputw()`*	Writes only integer to the file *`Ex: putw(n,fp);`*
`fgets()`	Reads a string from the a file *`Ex: fgets(ch,sizeof(string),fp);`*
`fputs()`	`Writes a string to a file` *`Ex: fputs(ch,fp);`*
`fseek()`	Sets the position to a desired point in the file *`Ex: fseek(fp,0,SEEK_SET);`*
`ftell()`	Gives the current position in the file *`Ex: n=ftell(fp);`*

Opening files: fopen( )

- ✓ fopen() is a predefined file handling function which is used to open already existing file or to create a file.
- ✓ It accepts two strings, the first is the name of the file, and the second is the mode in which it should be opened.

Syntax:

*FILE *filepointer; filepointer=fopen("filename.filetype","mode");*

Example:

```
FILE *fp;
fp= fopen("gitam.txt","w");
```

where gitam.txt is file name.

where 'w' is write mode.

Note: The function fopen( ) is compulsory for the files to do any operations, without fopen( ), we cannot perform operations on files.

S.No.	Mode	Description
1.	"r"	✓ Open for reading. ✓ The precondition is file must exist. ✓ Otherwise function returns Null Value.
2.	"w"	✓ Open for writing. ✓ If file already exists, its contents are overwritten. ✓ Otherwise a new file is created. ✓ The function returns NULL if it is unable to create the file. ✓ File cannot created for the reasons such as not having access permission, disk full, having write protect etc.,.

3.	"a"	✓ Open for appending. ✓ If file already exist, the new contents are appended. ✓ Otherwise, a new file is created. ✓ The function return NULL if it is unable to create the file.
4.	"r+"	✓ Open for both reading and writing. ✓ The file must exists.
5.	"w+"	✓ Open for both reading and writing. ✓ Contents written over.
6.	"a+"	✓ Open for reading and appending. ✓ If file is not existing the file is created.

Where will be file saved in computer?

A) If you didn't mention the path, by default the file will be saved in TC folder, i.e., in the drive where C-Lang is installed.

Example: fp=fopen("hello.txt", "w");

B) If you mention the path in fopen( ) function, then the new file will be saved in that particular path. Path should be mentioned as follows:

Example: fp=fopen("d:\\gitam\\hello.txt", "w");

(Or)

fp=fopen("d:/gitam/hello.txt", "w");

Writing to Files : fprintf( )

- ✓ It is a predefined file handling function which is used to print the content in a file.
- ✓ The fprintf( ) function is identical to printf( ) function except that they work on files.
- ✓ The first argument of this function is a file pointer which specifies the file to be used.

Syntax: *fprintf(fp,"control string", var_list);*

Where **fp** id is a file pointer associated with a file that has been opened for writing. The control string is file output specifications list may include variable, constant and string.

Example:

fprintf(fp, "%s%d%f",name,age,weight);

Here, **name** is an array variable of type char and **age** is an int variable

Reading from Files : fscanf( )

- ✓ It is a predefined file handling function which is used to read the content from a file.
- ✓ The fscanf( ) function is identical to scanf( ) function except that they work on files.
- ✓ The first argument of this function is a file pointer which specifies the file to be used.

Syntax: *fscanf(fp, "controlstring",var_list);*

This statement would cause the reading of items in the control string.

Example:

fscanf(fp, "%s%d",name,&quantity");

Where **fp** id is a file pointer associated with a file that has been opened for reading. The control string is file output specifications list may include variable, constant and string.

Closing the File : fclose(fp)

- ✓ When we have finished reading from the file, we need to close it. This is done using the function **fclose** through the statement, fclose(fp);

Note:

- ✓ During a write to a file, the data written is not put on the disk immediately. It is stored in a buffer. When the buffer is full, all its contents are actually written to the disk. The process of emptying the buffer by writing its contents to disk is called flushing the buffer.
- ✓ Closing the file flushes the buffer and releases the space taken by the FILE structure which is returned by fopen. Operating System normally imposes a limit on the number of files that can be opened by a process at a time. Hence, closing a file means that another can be opened in its place. Hence, if a particular FILE pointer is not required after a certain point in a program, pass it to fclose and close the file.

Syntax: *fclose(filepointer);*

Example:

fclose(fp); // Where fp is a file pointer

Example Program 01:Using write mode to create new file and to write the data in file.

```
#include<stdio.h>
void main()
{
    int i;
    char name[15];
    FILE *fp;  fp=fopen("data1.txt","w");
    printf("Input an integer");
    scanf("%d",&i);
    printf("Enter your name:");
    scanf("%s",name);
    fprintf(fp,"%d\n%s",i,name);
    fclose(fp);
}
```

Output:

```
Input an integer 5
Enter ur name:gitam
Data1.txt
5 gitam
```

Example Program 02: Using read mode to read the data from file.

```
#include<stdio.h>
void main()
{
   int i;
   char name[15];
   FILE *fp;
   fp=fopen("data1.txt","r");
   fscanf(fp,"%d%s",&i,name);
   printf("The integer in data1.txt is %d\n",i);
   printf("The String in data1.txt is %s",name);
   fclose(fp);
}
```

Output:

```
The integer in data1.txt is 5
The String in data1.txt is gitam
```

Example Program 03: Using append mode to add the data in the existing file.

```
#include<stdio.h>
void main()
{
    char name[15];
    FILE *fp;
    fp=fopen("data2.txt","a");
    printf("Enter ur name:");
    scanf("%s",name);
    fprintf(fp,"%s",name);
    fclose(fp);
    fp=fopen("data2.txt","r");
    if(fscanf(fp,"%s",name))
    printf("%s",name);
    fclose(fp);
}
```

Output:

```
Enter ur name:
     GITAM
     GITAM
     In data2.txt
     GITAM
Again after appending:
     Enter Ur name: University
     GITAMUniversity In data2.txt
     GITAMUniversity
```