**Files**

A file is a collocation of data stored on a secondary storage device like hard disk.

We can input data dynamically to the file by programming.

In a text file, each file ends with a special character called the end-of-file (EOF , EOF=1) maker.

Let’s take an example of .txt file we to make changes first we have to open the file then do the necessary changes and then save and close it. Same steps we are going to follow to do it by code.

To use files in c, we must follow these steps

* Declaration of file pointer variable
* Open the file
* Process the file
* Close the file

**Declaring a file pointer variable:**

To declare a file pointer we have to use a function **FILE** (All are in uppercase letter). The most important thing is we must use pointers to access the files if we use filename rather than file pointer then error will be pupped up.

Syntax:

FILE \*file\_pointer\_name;

Here file\_pointer\_name is a file pointer.

**Opening a file using fopen()**

To open any file there is a function works named **fopen()**

Prototype:

FILE \*fopen(const char \*file\_name/file\_path, const char \*mode);

If successful fopen() returns a pointer-to-structure and if it fails, it returns NULL.

The file path gives the information about location of the file on the disk.

Ex: path c:\filename\subfile\_name.extension

The first argument contains file path pointer and the second contains file mode. Mode conveys to c the type of processing data will be done with the file. There are different types of modes in c.

|  |  |
| --- | --- |
| r | To open a text file for reading. If the file does not exist it will give error. |
| w | To open a text file for writing. If the stream(file) not exist then it is created. If already exist then content would be deleted. |
| a | Append to a text file. If the file does not exist then created. |
| rb | Open a binary file for reading. b indicates binary |
| wb | Open a binary file for writing. |
| ab | Append to a binary file. |
| r+ | Open a text file for both reading and writing. |
| w+ | Open a text file for both reading and writing. The stream will be created if it does not exist, and will be truncated if it exists. |
| a+ | Open a text file for both reading and writing. The stream will be positioned at the end of the file content. |
| r+b / rb+ | Open a binary file for read/write. |
| w+b / wb+ | Open a binary file for read/write. |
| a+b / ab+ | Open a binary file for read/write. |

Ex: //WAP to open a file using fopen() function.

#include<stdio.h>

main()

{

int ch;

FILE \*fp;

fp=fopen("C:/Users/Lenovo/Desktop/Himansu/My project/applanding.html","r");

if(fp==NULL)

{

printf("The file could not be opened!");

}

else

{

while((ch=fgetc(fp)) != EOF)

printf("%c",ch);

}

}

**Closing a file using fclose()**

To close an opened file the fclose() function is used. Which disconnects file pointer from a file. After the fclose() disconnects the file pointers can be used to access a different file or the same but in different mode.

Syntax:

int fclose(FILE \*fp);

If success returns 0. If unsuccess returns non-zero value.

In addition to fclose() there is another functions **fcloseall()** which closes all the streams that are currently opened except the standard streams (such as stdin, stdout and stderr).

Syntax: int fcloseall(void);

If a file buffer has to be flushed without closing it then use **fflush()** or **flushall()** to flush the buffers of all open streams.

**Read data from a file:**

C provides curtain functions to read data from a file.

1. fscanf()
2. fgets()
3. fgetc()
4. fread()

**fscanf()** :

It is used to read formatted data from a string.

Syntax: int fscanf(FILE \*stream, const char \*format, ……)

This is used to read data from stream and store them according to the parameter format into the location pointed by the additional arguments.

Ex:

//WAP to demonstrate the use of fscanf() and fclose().

#include<stdio.h>

int main()

{

FILE \*fp;

int roll\_no; char name[50];

fp=fopen("C:/Users/chiku/chiku.txt","r");

if(fp==NULL)

{

printf("The file couldn't be opened!");

exit(1);

}

fscanf(fp,"%s %d",name,&roll\_no); //it will take data from file

printf("\n Name:%s\tRoll No:%d",name,roll\_no);

fclose(fp);

return 0;

}

**fgets()** :

This stands for file get string. This is used to get a string from a stream.

Syntax: char \*fgets(char \*str, int size, FILE \*stream);

On success it returns the string pointer str.

The fgets() function reads at most size-1 number of character from the stream and stores them into string pointer str.

fgets() terminates as soon as in finds newline character, EOF or any other error. As it a string reading function after read without any error a ‘ \0 ’ character is appended to the end of the string.

The gets() functions and fgets() functions are almost same except gets() has an infinite size and a stream of stdin. Another difference gets() never retains newline character but fgets() keep it.

Ex:

// WAP to demonstrate the use of fgets() .

#include<stdio.h>

int main()

{

FILE \*fp;

char str[50];

fp = fopen("C:/Users/chiku/chiku.txt","r");

if(fp==NULL)

{

printf("The file couldn't be opened!");

exit(1);

}

while(fgets(str,50,fp) != NULL)

{

printf("%s ",str); //This file will read 49 character one by one

}

fclose(fp);

return 0;

}

**fgetc() :**

This functions returns the next character from the stream. EOF is the end if the file is reached.

Syntax: int fgetc(FILE \*stream);

fgetc() returns the character read as an int on return EOF to indicate an error or end of the file.

Ex:

//WAP to demonstrate the use of fgetc() function in file.

#include<stdio.h>

int main()

{

FILE \*fp;

int ch; char str[90];

fp = fopen("C:/Users/chiku/chiku.txt","r");

if(fp==NULL)

{

printf("The file couldn't be opened!");

exit(1);

}

int i=0;

ch = fgetc(fp); /\* we can also use this concept\*/

while(i<89 && (feof(fp)==0)) // while( (ch=(fgetc(fp))) != EOF)

{ // {

str[i] = (char)ch; // str[i] = (char)ch; i++; }

ch = fgetc(fp);

i++;

}

str[i]='\0';

printf("%s",str);

fclose(fp);

return 0;

}

**fread() :**

It is used to read data from file.

Syntax: int fread(void \*str, size\_t size, size\_t num, FILE \*stream);

On success it returns the number of bytes successfully read.

**Writing data to files:**

C provides the following set of function to write data in a file.

1. fprintf()
2. fputs()
3. fputc()
4. fwrite()

**fprintf() :**

This is used to write formatted output stream.

Syntax: int fprintf(FILE \*stream,const char \*format,……);

Ex:

// WAP to demonstrate the use of fprintf() function.

#include<stdio.h>

int main()

{

FILE \*fp;

int i; char name[20];

float salary;

fp= fopen("C:/Users/chiku/chiku.txt","w");

if(fp==NULL)

{

printf("The could not be opened!");

exit(1);

}

for(i=0;i<10;i++)

{

puts("\nEnter your name: ");

gets(name);

fflush(stdin);

puts("\nEnter your salary: ");

scanf("%f",&salary);

fprintf(fp,"%d Name %s \t salary %f",i,name,salary);

}

fclose(fp);

return 0;

}

**fputs() :**

It is opposite to fgets(). The fputs() is used to write a line to a file.

Syntax: int fputs(const char \*str, FILE \*stream);

It writes the string pointed to str by str to the stream pointed by stream.

On successfully completion it returns 0. In case of any error it returns EOF.

Ex:

// WAP to demonstrate the use of fputs() function.

#include<stdio.h>

int main()

{

FILE \*fp;

char feedback[40];

fp= fopen("C:/Users/chiku/chiku.txt","w");

if(fp==NULL)

{

printf("The could not be opened!");

exit(1);

}

puts("Give a feedback!");

gets(feedback);

fflush(stdin);

fputs(feedback,fp);

printf("%s",feedback);

fclose(fp);

return 0;

}

**fputc() :**

It is just opposite of fgetc() and used to write a character to the stream.

Prototype: int fputc(int ch, FILE \*stream);

On success it returns the value it has written. On failure the function will return EOF and the error indicator for the stream will be set.

Ex:

// WAP to demonstrate the use of fputc() function.

#include<stdio.h>

int main()

{

FILE \*fp;

char feedback[40];

int ch,i;

fp= fopen("C:/Users/chiku/chiku.txt","w");

if(fp==NULL)

{

printf("The could not be opened!");

exit(1);

}

puts("Give an important feedback!");

gets(feedback);

for(i=0;i<feedback[i];i++)

fputc(feedback[i],fp);

printf("%s",feedback);

fclose(fp);

return 0;

}

**fwrite() :**

It is used to write data to a file.

Syntax: int fwrite(const void \*str, size\_t size, size\_t count, FILE \*stream);