

56. File Management In C Programming:

Introduction:

File defines a computer resource for recording data in a computer storage device, primarily identified by its file name. Just as words can be written to paper, so can data be written to a computer file. Files can be shared with and transferred between computers and mobile devices via removable media, networks or the internet.

A file container in a computer system that stores data, information, settings or commands which are used with a computer program. In graphical user interface such as Microsoft operating systems, represent the files as icons, which associate to the program that opens the file. For instance the picture is shown as an icon, it is related to Microsoft Word. If your computer contains the file and you double-click on the icon, it will open in Microsoft Word installed on the computer. A file can be defined as a collection of characters stored on a permanent storage device. C programming language treats all the devices as files: devices as standard output, printer etc. C programming language provides a rich set of built-in functions to read given input and feed it to the program as per requirement as well as for output.

In programming we may require some specific input data to be generated several numbers of times. Sometimes it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling in C. 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.

- | | | |
|-----------------------------|-----------------------------|----------------------|
| 1. Creation Of The New File | 2. Opening An Existing File | |
| 3. Reading From The File | 4. Writing To The File | 5. Deleting The File |

The following files are automatically opened when a program executes to provide access to the keyboard (console input device) and screen (console output device)

Standard input – stdin – keyboard

Standard output – stdout – screen

Standard error – stderr – your screen

Functions For File Handling In C Programming:

These are the following functions that are used in C library to open, read, write, search and close file.

`fopen()` – open new or existing file

`fprintf()` – write data into the file

`fscanf()` – reads data from the file

`fputc()` – writes a character into the file

`fgetc()` – reads a character from file

`fputs()` – writes an String to file

`fgets()` – reads an String from file

`fseek()` – sets the file pointer to given position

`ftell()` – returns current position

`rewind()` – sets the file pointer to the beginning of the file

`fclose()` – close the file

Opening File: fopen()

`fopen()` function is used for open new or existing file. You can use the `fopen()` function to create a new file or to open an existing file, this call will initialize an object of the type `FILE`, which contains all the information necessary to control the stream. “`FILE *fopen(const char * filename, const char * mode);`” The file name (string). If the file is stored at some specific location, then we must mention the path at which the is stored. The mode in which the files is to be opened. It is a string. The following modes are used in `fopen()` function.

`r` = opens a text file in read mode

`r+` = open a text file in read and write mode

`w` = opens a text file in write mode

`w+` = open a text file in read and write mode

`a` = opens a text file in append mode

`a+` = open a text file in read and write mode

rb = opens a binary file in read mode

rb+ = opens a binary file in read and write mode

wb = opens a binary file in write mode

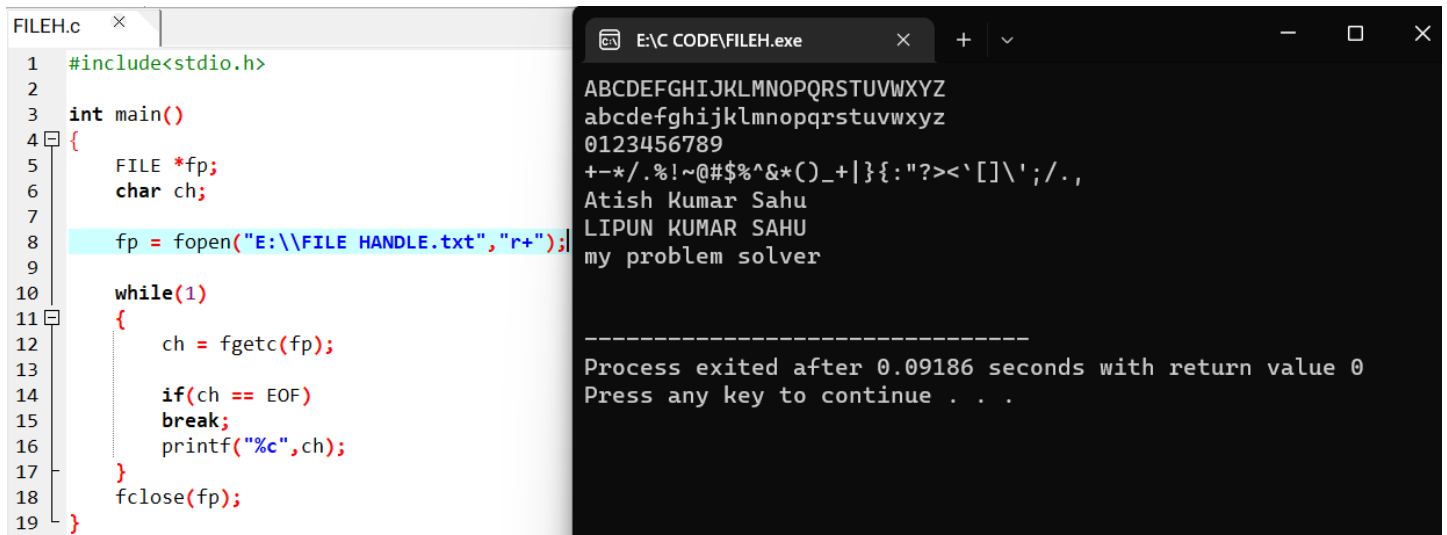
wb+ = opens a binary file in read and write mode

ab = open a binary file in append mode

ab+ = opens a binary file in read and write mode

fopen function works in the following way:

firstly it searches the file to be opened. Then it loads the file from the disk and place it into the buffer. The buffer is used to provide efficiency for the read operations. It sets up a character pointer which points to the first character of the file.

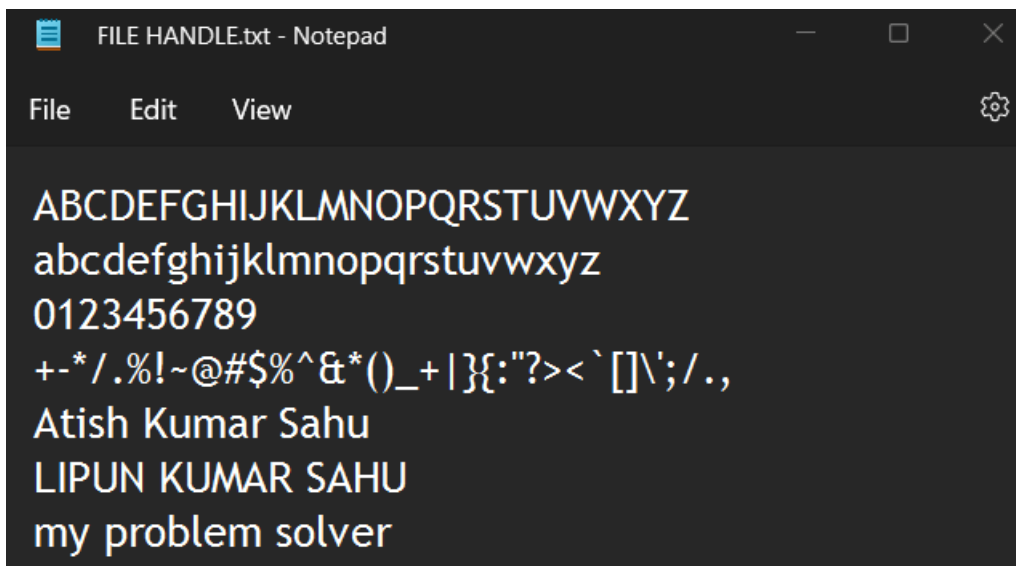


The image shows a C program in a file named FILEH.c and its execution output in a terminal window. The program opens a file named "E:\\FILE HANDLE.txt" in read mode ("r+"). It then reads the file character by character using fgetc and prints each character on a new line using printf. The file contains the following text: ABCDEFGHIJKLMNOPQRSTUVWXYZ, abcdefghijklmnopqrstuvwxyz, 0123456789, a line of special characters, "Atish Kumar Sahu", "LIPUN KUMAR SAHU", and "my problem solver". The terminal output matches the file content. Below the file content, the terminal shows a message: "Process exited after 0.09186 seconds with return value 0" and "Press any key to continue . . .".

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *fp;
6     char ch;
7
8     fp = fopen("E:\\FILE HANDLE.txt", "r+");
9
10    while(1)
11    {
12        ch = fgetc(fp);
13
14        if(ch == EOF)
15            break;
16        printf("%c", ch);
17    }
18    fclose(fp);
19 }
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
+~*./.%!~@#$$%^&*()_+|}{:~?><`[]\';/.,
Atish Kumar Sahu
LIPUN KUMAR SAHU
my problem solver

-----
Process exited after 0.09186 seconds with return value 0
Press any key to continue . . .
```



The image shows a Notepad window titled "FILE HANDLE.txt - Notepad". The window contains the same text as the file shown in the previous screenshot: ABCDEFGHIJKLMNOPQRSTUVWXYZ, abcdefghijklmnopqrstuvwxyz, 0123456789, a line of special characters, "Atish Kumar Sahu", "LIPUN KUMAR SAHU", and "my problem solver".

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
+~*./.%!~@#$$%^&*()_+|}{:~?><`[]\';/.,
Atish Kumar Sahu
LIPUN KUMAR SAHU
my problem solver
```

fprintf() & fscanf():

fprintf() function is used to write set of characters into file. It sends formatted output to a stream. "int fprintf(FILE *stream, const char *format [argument,...];"

fscanf() function is used to read set of characters from file. it reads a word from the file and returns EOF(End Of File) at the end of file.

```
FILEH.c  x
1  #include<stdio.h>
2
3  int main()
4  {
5      FILE *fp;
6      fp = fopen("E:\\FILE HANDLE1.txt","w");
7      fprintf(fp,"This file is created by using C language.\n");
8      fprintf(fp,"This Is A Demo Text For Testing Purpose");
9      fclose(fp);
10 }
```

FILE HANDLE1.txt - Notepad

File Edit View

This file is created by using C language.
This Is A Demo Text For Testing Purpose

```
FILEH.c  x
1  #include<stdio.h>
2
3  int main()
4  {
5      FILE *fp;
6      char read[200];
7
8      fp = fopen("E:\\FILE HANDLE1.txt","r");
9
10 while(fscanf(fp,"%s",read) != EOF)
11 {
12     printf("%s",read);
13 }
14 fclose(fp);
15 }
```

E:\C CODE\FILEH.exe

ThisfileiscreatedbyusingClanguage.ThisIsADemoTextForTestingP
urpose

Process exited after 0.06288 seconds with return value 0
Press any key to continue . . .

```
FILEH.c x
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *fcrt;
6     fcrt = fopen("E:\\EMPFILE.txt", "w+");
7
8     int empid;
9     char empname[100];
10    float salary;
11
12    printf("Enter The Empid : ");
13    scanf("%d",&empid);
14    fprintf(fcrt, "Emp Id : %d\n", empid);
15
16    printf("Enter The Emp Name : ");
17    scanf("%s", empname);
18    fprintf(fcrt, "Emp Name : %s\n", empname);
19
20    printf("Enter The Emp Salary : ");
21    scanf("%f",&salary);
22    fprintf(fcrt, "Emp Salary : %f\n", salary);
23
24    fclose(fcrt);
25
26    FILE *fread;
27    fread = fopen("E:\\EMPFILE.txt", "r");
28    char read[500];
29    while(fscanf(fread, "%s", read) != EOF)
30    {
31        printf("%s\n", read);
32    }
33    fclose(fread);
34 }
```

```
E:\C CODE\FILEH.exe x + v
Enter The Empid : 101
Enter The Emp Name : Atish
Enter The Emp Salary : 50000.0
Emp
Id
:
101
Emp
Name
:
Atish
Emp
Salary
:
50000.000000
-----
Process exited after 12.6 seconds with return value 0
Press any key to continue . . .
```

```
EMPFILE.txt - Notepad
File Edit View
Emp Id : 101
Emp Name : Atish
Emp Salary : 50000.000000
```

fputc() & fgetc():

The fputc() function is used to write a single character into file. it outputs a character to a stream. "int fputc(int c, FILE *STREAM);"

The fgetc() function returns a single character from the file. it gets a character from the stream. It returns EOF at the end of the file. "int fgetc(FILE *stream);"

```
FILEH.c x
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file;
6     file = fopen("E:\\putcgetc.txt", "w+");
7     fputc('A', file);
8     fputc('a', file);
9     fputc('K', file);
10    fputc('k', file);
11    fputc('S', file);
12    fputc('s', file);
13    fclose(file);
14 }
```

```
putcgetc.txt - Notepad
File Edit View
AaKkSs
```

The screenshot shows a C program in a file named FILEH.c. The code includes `<stdio.h>` and defines a `main()` function. Inside `main()`, a file named `E:\\putcgetc.txt` is opened in read mode (`"r+"`). A `while` loop reads characters from the file using `fgetc(file1)` until it reaches the end of the file (`EOF`). Each character is printed to the console using `printf("%c", ch);`. After the loop, the file is closed with `fclose(file1);`. The output window shows the characters 'AaKkSs' followed by a separator line, then a message indicating the process exited after 0.06309 seconds with a return value of 0, and a prompt to press any key to continue.

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file1;
6     char ch;
7     file1 = fopen("E:\\putcgetc.txt", "r+");
8
9     while((ch = fgetc(file1)) != EOF)
10    {
11        printf("%c", ch);
12    }
13    fclose(file1);
14 }
```

AaKkSs

Process exited after 0.06309 seconds with return value 0
Press any key to continue . . .

fputs() & fgets():

The `fputs()` function writes a line of character into file. it outputs string to a stream. "int `fputs(const char *s, FILE *stream);`"

The `fgets()` function reads a line of character from file. It gets String from a stream. "char* `fgets(char *s, int n, FILE *stream);`"

The screenshot shows a C program in a file named FILEH.c. The code includes `<stdio.h>` and defines a `main()` function. Inside `main()`, a file named `E:\\PUTSGETS.txt` is opened in write mode (`"w+"`). A string `"This Is A txt File. Welcome to C programming"` is written to the file using `fputs(string, file);`. The file is then closed with `fclose(file);`. To the right, a Notepad window shows the content of `PUTSGETS.txt`, which is `This Is A txt File. Welcome to C programming`.

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file;
6     file = fopen("E:\\PUTSGETS.txt", "w+");
7
8     fputs("This Is A txt File. Welcome to C programming", file);
9
10    fclose(file);
11 }
```

PUTSGETS.txt - Notepad
File Edit View
This Is A txt File. Welcome to C programming

The screenshot shows a C program in a file named FILEH.c. The code includes `<stdio.h>` and defines a `main()` function. Inside `main()`, a file named `E:\\PUTSGETS.txt` is opened in read mode (`"r+"`). A character array `ch` of size 200 is declared. The string from the file is read into `ch` using `fgets(ch, 200, file1);`. The string is then printed to the console using `printf("%s", ch);`. The file is closed with `fclose(file1);`. The output window shows the string `This Is A txt File. Welcome to C programming` followed by a separator line, then a message indicating the process exited after 0.1401 seconds with a return value of 0, and a prompt to press any key to continue.

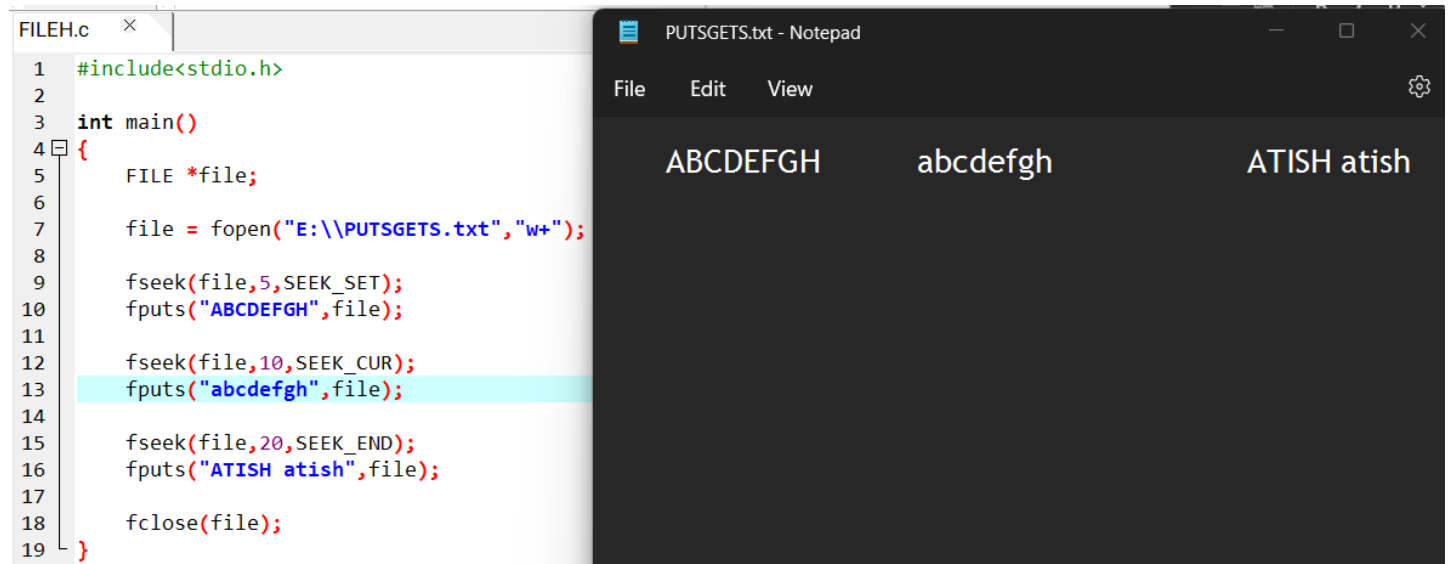
```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file1;
6     char ch[200];
7     file1 = fopen("E:\\PUTSGETS.txt", "r+");
8     printf("%s", fgets(ch, 200, file1));
9     fclose(file1);
10 }
```

This Is A txt File. Welcome to C programming

Process exited after 0.1401 seconds with return value 0
Press any key to continue . . .

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. "int fseek(FILE *stream,long int offset, int whence);" there are 3 constants used in the fseek() function for the whence those are "SEEK_SET, SEEK_CUR, SEEK_END".



The screenshot displays a C program in FILEH.c and its output in a Notepad window. The C program uses fseek() to move the file pointer to the beginning (SEEK_SET), then to the current position (SEEK_CUR), and finally to the end (SEEK_END) of the file. The output in the Notepad window shows the data written at each step: "ABCDEFGH", "abcdefgh", and "ATISH atish".

```
FILEH.c
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file;
6
7     file = fopen("E:\\PUTSGETS.txt","w+");
8
9     fseek(file,5,SEEK_SET);
10    fputs("ABCDEFGH",file);
11
12    fseek(file,10,SEEK_CUR);
13    fputs("abcdefgh",file);
14
15    fseek(file,20,SEEK_END);
16    fputs("ATISH atish",file);
17
18    fclose(file);
19 }
```

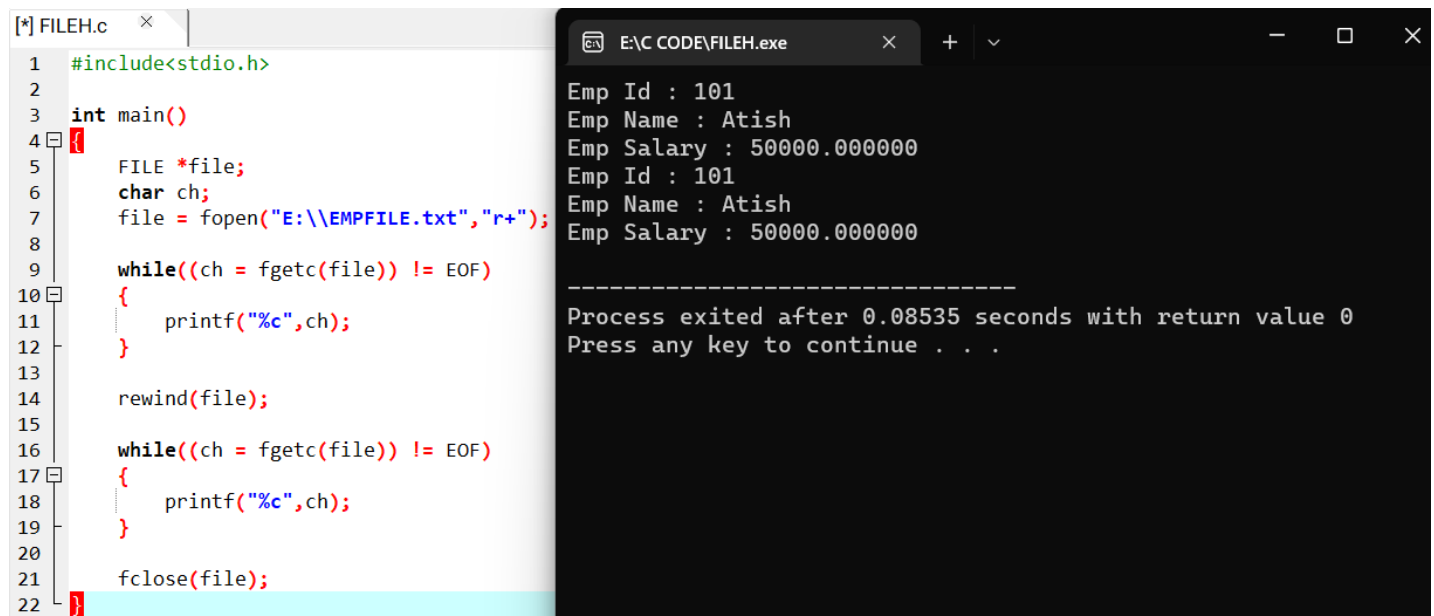
PUTSGETS.txt - Notepad

File Edit View

ABCDEFGH abcdefgh ATISH atish

rewind() function:

The rewind() function sets the file pointer at beginning of the stream. it is useful if you have to stream many times. "void rewind(FILE *stream);"



The screenshot displays a C program in FILEH.c and its output in a command prompt window. The C program uses rewind() to reset the file pointer to the beginning of the stream, allowing the data to be read again. The output in the command prompt window shows the data read from the file: "Emp Id : 101", "Emp Name : Atish", and "Emp Salary : 50000.000000".

```
[*] FILEH.c
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file;
6     char ch;
7     file = fopen("E:\\EMPFILE.txt","r+");
8
9     while((ch = fgetc(file)) != EOF)
10    {
11        printf("%c",ch);
12    }
13
14    rewind(file);
15
16    while((ch = fgetc(file)) != EOF)
17    {
18        printf("%c",ch);
19    }
20
21    fclose(file);
22 }
```

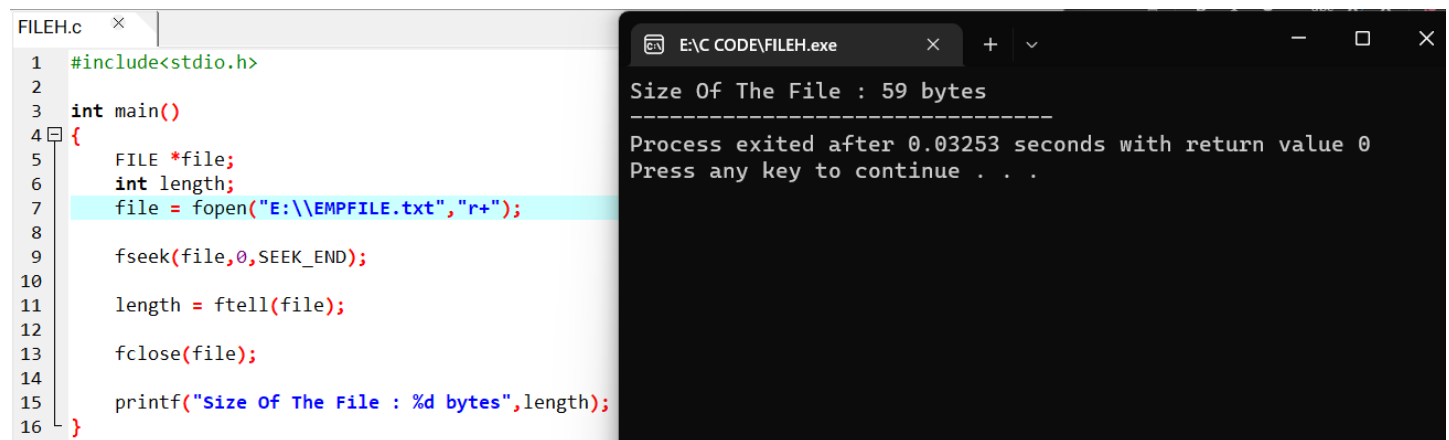
E:\C CODE\FILEH.exe

Emp Id : 101
Emp Name : Atish
Emp Salary : 50000.000000
Emp Id : 101
Emp Name : Atish
Emp Salary : 50000.000000

Process exited after 0.08535 seconds with return value 0
Press any key to continue . . .

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. “`long int ftell(FILE *stream);`”



The image shows a C program in a text editor and its execution output in a terminal window. The program, named `FILEH.c`, includes `<stdio.h>` and defines a `main()` function. Inside `main()`, it declares `FILE *file;` and `int length;`. It then opens a file named `"E:\\EMPPFILE.txt"` in append mode using `fopen("E:\\EMPPFILE.txt", "r+");`. The file pointer is moved to the end of the file using `fseek(file, 0, SEEK_END);`. The current file position is then obtained using `length = ftell(file);`. Finally, the file is closed with `fclose(file);` and the size is printed using `printf("Size Of The File : %d bytes", length);`.

The terminal window, titled `E:\\C CODE\\FILEH.exe`, shows the output of the program: `Size Of The File : 59 bytes`, followed by a separator line of dashes. Below this, it states `Process exited after 0.03253 seconds with return value 0` and `Press any key to continue . . .`.

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *file;
6     int length;
7     file = fopen("E:\\EMPPFILE.txt", "r+");
8
9     fseek(file, 0, SEEK_END);
10
11     length = ftell(file);
12
13     fclose(file);
14
15     printf("Size Of The File : %d bytes", length);
16 }
```

```
Size Of The File : 59 bytes
-----
Process exited after 0.03253 seconds with return value 0
Press any key to continue . . .
```