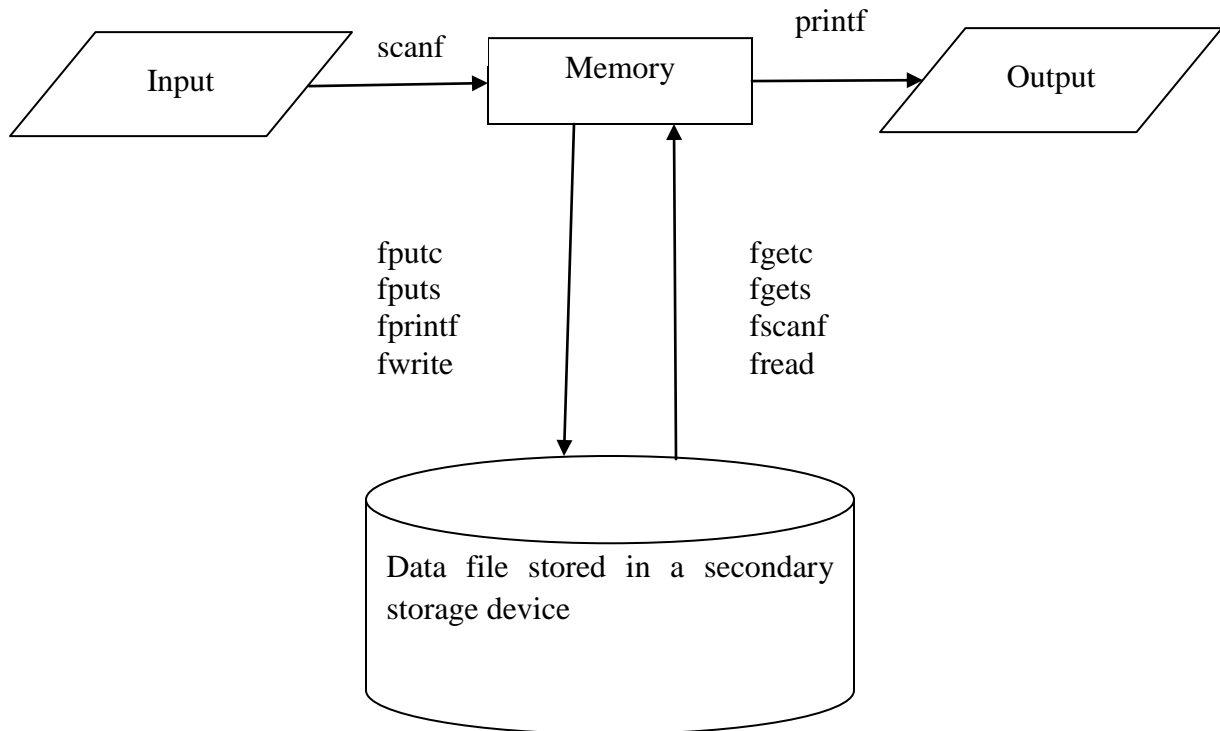# Unit-10

Files and File Handling in C

Concept of file

File is a collection of related data placed on the disk. C support the concept of files through which data can be stored on the disk or secondary storage device. The stored data can be read whenever required.

Opening a File

A file must be opened before any I/O operations performed on that file. The process of establishing a connection between the program and file is called opening the file. A structure named FILE is defined in the file stdio.h that contains all information about the file like name, status, buffer size, current position, end of file status etc. All these details are hidden from the programming and the operating system takes care of all these things.

A file pointer is a pointer to a structure of type FILE. Whenever a file is opened, a structure of type FILE is associated with it and a file pointer that points to this structure identifies this file.

Eg:

```
FILE *fp;
```

The function fopen ( ) is used to open a file.

Eg:

```
fp=fopen("path:\\filename.ext","file_operation_mode");
```

Closing of file

The file that was opened using fopen ( ) function must be closed when no more operations are to be performed on it. After closing the file, connection between file and program is broken.

On closing the file, all the buffers associated with it are flushed i.e all the data that is in the buffer is written to the file. The buffers allocated by the system for the files are freed after the file is closed, so that these buffers can be available for other files.

Although all the files are closed automatically when the program terminates, but sometimes it may be necessary to close the file by using fclose( ) function.

Modes

The different file modes that can be used in opening file are:

1. "w" (write)	If the file doesn't exist then this mode creates a new file for writing, and if the file already exists then the previous data is erased and the new data entered is written to the file.
2. "a" (append)	If the file doesn't exist then this mode creates a new file and if the file already exists then the new data entered is appended at the end of existing data. In this mode, the data existing in the file is not erased as in "w" mode.
3. "r" (read)	This mode is used for opening an existing file for reading purpose only. The file to be opened must exist and the previous data of the file is not erased.
4. "w+" (write + read)	This mode is same as "w" mode but in this mode we can also read and modify the data. If the file doesn't exist then a new file is created and if the file exists then previous data is erased.
5. "r+" (read + write)	This mode is same as "r" mode but in this mode we can also write and modify existing data. The file to be opened must exist and the previous data of file is not erased. Since we can add new data and modify existing data so this mode is also called updata mode.
6. "a+" (append + read)	This mode is same as the "a" mode but in this mode we can also read the data stored in the file. If the file doesn't exist, a new file is created and if the file already exists then new data is appended at the end of existing data. We cannot modify existing data in this mode.
7. "wb"	Binary file opened in write mode.
8. "ab"	Binary file opened in append mode.
9. "rb"	Binary file opened in read mode.
10. "wb+"	Create a binary file for read/write.
11. "rb+"	Open a binary file for read/write.
12. "ab+"	Append a binary file for read/write.

Text mode
- In text mode every digit or text are stored as a character and while reading the content back, the conversion is required from character to appropriate format and takes lots of space.
- Character I/O, string I/O, and formatted I/O use text mode.
- If 3.14159 is stored in character mode file size would be 8 bytes (counts each character including decimal and EOF).

Binary mode
- In binary mode every digit or text is stored in binary format and while reading the content no conversion is necessary and takes little space.
- fread ( ) and fwrite ( ) are used in binary mode.
- If 3.14159 is stored in character mode file size would be 4 bytes.

Input/Output Functions

The functions used for file input/output are

1. Character I/O
   a. fputc( )
      This function writes a character to the specified file at the current file position and then increments the file position pointer.
   b. fgetc( )
      This function reads a single character from a given file and increments the file pointer.
   c. getc( ) and putc( )
      The operation of getc( ) and putc( ) are exactly similar to that of fgetc( ) and fputc( ) , the only difference is that the former two are defined as macros while the latter two are functions.
2. Integer I/O
   a. putw( )
      This function writes an integer value to the file pointed to by file_pointer.
   b. getw( )
      This function returns the integer value from the file associated with file_pointer.
3. String I/O
   a. fputs( )
      This function writes the null terminated string pointed by given character pointer to a file.
   b. fgets()
      This function is used to read characters from a file and these characters are stored in the string pointed by a character pointer.
4. Formatted I/O
   a. fprintf( )
      This function is same as the printf( ) function but it writes formatted data into the file instead of the standard output(screen). This function has same parameters as in printf( ) but it has one additional parameter which is a pointer of FILE type, that points to the file to which the output is to be written.
   b. fscanf( )
      This function is similar to the scanf( ) function but it reads data from file instead of standard input, so it has one more parameter which is a pointer of FILE type and it points to the file from which data will be read.
5. Block Read/Write
   a. fwrite( )
      This function is used for writing an entire block to a given file.
   b. fread( )
      This function is used to read an entire block from a given file.

Random file processing

In this technique the content are stored sequentially and read back the content from any position of the file. And it can be performed using some function like rewind( ), fseek( ) and ftell( ).

rewind( )
The rewind( ) function places the file pointer to the beginning of the file.
rewind(file_pointer);

fseek( )
This function is used for setting the file position pointer at the specified byte.
fseek(file_pointer,offset,offset_initial);

ftell( )
This function returns the current position of the file position pointer. The value is counted from beginning of the file.
position=ftell(fp);   where position is long int.

➢ Write a program to write name, symbolno and address of 'n' students into a file named "student.txt" and display the record of students in appropriate format.

```c
#include<stdio.h>
#include<conio.h>
void main()
{
struct student
{
int sno;
char name[15];
char add[15];
};
struct student st;
FILE *fp;
fp=fopen("student.txt","w+");
int i,n;
clrscr();
printf("How many students are there: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter symbolno, name and address of student: ");
scanf("%d%s%s",&st.sno,st.name,st.add);
fwrite(&st,sizeof(st),1,fp);
}
rewind(fp);
printf("\nSymbno\tName\tAddress");
while(fread(&st,sizeof(st),1,fp)==1)
{
printf("\n%d\t%s\t%s",st.sno,st.name,st.add);
}
fclose(fp);
getch();
}
```

Output

```
How many students are there: 5

Enter symbolno, name and address of student: 13001 rabina gaindakot

Enter symbolno, name and address of student: 13002 padma butwal

Enter symbolno, name and address of student: 13003 rakesh kathmandu

Enter symbolno, name and address of student: 13004 mausham dhangadi

Enter symbolno, name and address of student: 13005 mahesh narayangarh

Symbno  Name    Address
13001   rabina  gaindakot
13002   padma   butwal
13003   rakesh  kathmandu
13004   mausham dhangadi
13005   mahesh  narayangarh_
```

➢ Write a program to delete and rename the file.
```c
#include<stdio.h>
#include<conio.h>
void main()
{
char str[20];
FILE *fp;
clrscr();
fp=fopen("fil.txt","w");
printf("\nInput string: ");
scanf("%s",str);
fprintf(fp,"%s",str);
fclose(fp);
rename("fil.txt","newfil.txt"); /* rename the file name to newfil.txt */
remove("newfil.txt");           /* removes(delete) file newfil.txt */
getch();
}
```
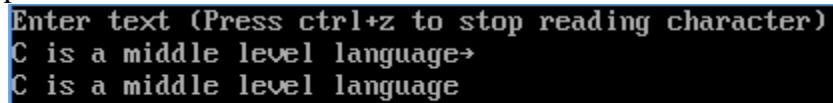➢ Write a program to copy contents of one file to another
```c
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp1,*fp2;
char ch;
clrscr();
fp1=fopen("abc.txt","r");
fp2=fopen("xyz.txt","w");
while((ch=getc(fp1))!=EOF)
{
putc(ch,fp2);
}
fclose(fp1);
fclose(fp2);
getch();
}
```
➢ Program to understand fputc( ) and fgetc( ).
```c
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
char ch;
clrscr();
fp=fopen("abc.txt","w");
printf("Enter text (Press ctrl+z to stop reading character)\n");
while((ch=getchar())!=EOF)
{
fputc(ch,fp);
}
fclose(fp);
fp=fopen("abc.txt","r");
while((ch=fgetc(fp))!=EOF)
{
printf("%c",ch);
}
fclose(fp);
```
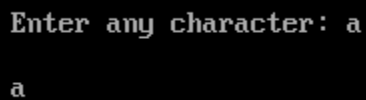
```
    getch();
    }
```

Output

> ➢ Program to understand putc( ) and getc( ).

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fptr;
char ch;
printf("\n Enter any character: ");
scanf("%c",&ch);
fptr=fopen("abc.txt","w");
putc(ch,fptr);
fclose(fptr);
fptr=fopen("abc.txt","r");
ch=getc(fptr);
printf("\n %c",ch);
getch();
}
```
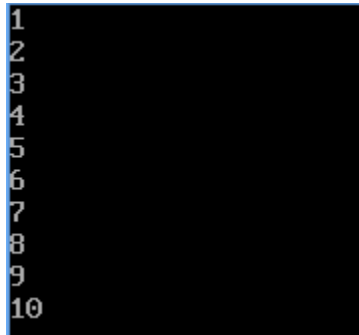
Output

> ➢ Program to understand putw( ) and getw( )

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
int value;
clrscr();
fp=fopen("abc.txt","w");
for(value=1;value<=10;value++)
{
putw(value,fp);
}
fclose(fp);
fp=fopen("abc.txt","r");
while((value=getw(fp))!=EOF)
{
printf("%d\n",value);
}
fclose(fp);
```
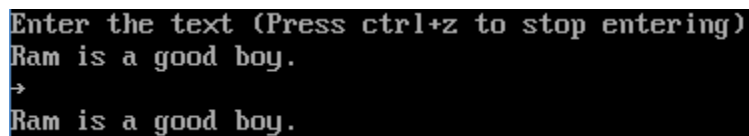
```
getch();
}
```

Output

➢ Program to understand fputs( ) and fgets( ).
```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fptr;
char str[80];
fptr=fopen("abc.txt","w");
printf("Enter the text (Press ctrl+z to stop entering) \n");
while(gets(str)!=NULL)
fputs(str,fptr);
fclose(fptr);
fptr=fopen("abc.txt","r");
while(fgets(str,80,fptr)!=NULL)
{
puts(str);
}
fclose(fptr);
getch();
}
```

Output

➢ Program to understand fprintf( ) and fscanf( )
```
#include<stdio.h>
#include<conio.h>
void main()
{
struct student
{
int roll;
char name[15];
};
struct student st;
FILE *fp;
```

```
int i,n;
clrscr();
printf("How many records do you want to enter: ");
scanf("%d",&n);
fp=fopen("student.txt","w");
for(i=0;i<n;i++)
{
printf("\n Enter rollno and name of student: ");
scanf("%d%s",&st.roll,st.name);
fprintf(fp,"\n%d%s",st.roll,st.name);
}
fclose(fp);
fp=fopen("student.txt","r");
printf("\n Rollno\tName");
while(fscanf(fp,"%d%s",&st.roll,st.name)!=EOF)
{
printf("\n %d\t%s",st.roll,st.name);
}
fclose(fp);
getch();
}
```

Output

```
How many records do you want to enter: 5

 Enter rollno and name of student: 2 mohan

 Enter rollno and name of student: 3 nabin

 Enter rollno and name of student: 4 raju

 Enter rollno and name of student: 5 mahesh

 Enter rollno and name of student: 8 deepak

 Rollno Name
 2       mohan
 3       nabin
 4       raju
 5       mahesh
 8       deepak_
```