



4.1 Introduction to files

C language provides a way to read or write the data from the secondary storage devices in the form of file.

A file is a collection of bytes stored on a secondary storage device, which is generally a disk.

The collection of bytes may be Characters, words, lines, paragraphs.



File is created for permanent storage of data. It is a readymade structure.

This way data can be stored permanently.

To work with files, C provides many build-in library functions.

A stream in C is a logical interface to the various devices such as keyboard, monitor, port, file etc. file stream I/O uses secondary storage devices for reading and writing data to a file.

So far, we have learned many input/output library functions such as scanf(), gets(), getch(), puts(), printf() etc. These functions operate on standard input device, keyboard and standard output device, console. In program, data are stored in variables and variables are a temporary storage. When program terminated the data stored in the variables is lost.

In processing a large amount data, it is time consuming and difficult to handle the data through input/output devices. We can store this data in a file and access it any time by few commands provided by C language.

The purpose of file handling is

- Data can be stored on disks and read whenever necessary.

We can perform various operations on files like

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



To Work with File we need to create reference as given below

Syntax#	Example#
FILE *pointername	FILE *ptr;





4.2 File operations

C language provides number of functions related with File operation.

Function	Description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

4.1.1 Fopen()

The fopen() function opens a file and creates an instance of the FILE structure and returns a pointer to that structure. This pointer is used with all subsequent file operations in the program.

Therefore, we must specify following things while dealing with file:

1. file name
2. pointer of FILE data structure
3. fopen() function



Syntax#	Example#
FILE *fopen(char *filename, char *mode);	FILE *fp; Fp = fopen("xyz", "w");

In the syntax, the name of file to be opened is pointed to by filename, how file is accessed is pointed by mode.

Filename may have a path that specifies the drive or directory location where file is stored. If the filename is specifying without a path, then file is assumed to be in the current directory. If the filename is specified with path than double backslash (\\) character is used to separate the directory. For example,D:\\jpp\\test.txt

File mode is used to specify what operation to be performed on file.

mode	Description
r	opens a text file in reading mode
w	opens or create a text file in writing mode.
a	opens a text file in append mode
r+	opens a text file in both reading and writing mode
w+	opens a text file in both reading and writing mode





a+	opens a text file in both reading and writing mode
rb	opens a binary file in reading mode
wb	opens or create a binary file in writing mode
ab	opens a binary file in append mode
rb+	opens a binary file in both reading and writing mode
wb+	opens a binary file in both reading and writing mode
ab+	opens a binary file in both reading and writing mode

Example#

Write mode	Read mode
FILE *fp; fp = fopen("c:\\1.txt" , "w");	FILE *fp; fp = fopen("c:\\1.txt" , "r");

If file does not open due to the following reason, NULL value is returned by fopen() function.

- If file name is not valid.
- If path (directory) of the file is not valid.
- If file is opened in a read mode and file does not exist.

We may verify these things by writing following lines of code.

Example# of File open

```
FILE *fp;
fp = fopen("abc.txt" , "w");
if(fp == NULL)
{
    printf("\n file creating error");
    exit(1);
}
```



4.1.2 Closing the file

After finishing the operations on the file, file should be closed. Because during the file operations file, pointer of FILE is associated with the file stream.

Therefore file should be closed using fclose() function.

Syntax#



Syntax#	Example#
int fclose(FILE *fp);	FILE *fp; fclose(fp);

It returns integer value either 0 on success or -1 on error during file close operation. When the program terminates, all the file streams are automatically flushed and closed.





4.3 Input/output operations on file

When dealing with files, files can be read or write. The way file is opened for processing, files streams are of type: character (text)file and binary file.

Character files stores every type of data in the form of character only. While binary files store the data in a various type such as int, float, char etc.

C provides various built-in library functions for dealing with the file.

There are two types of file Text and Binary file.



Difference between Text and Binary File

Text	Binary
Text File can only contain textual data.	Binary files typically contain a sequence of bytes, or ordered groupings of eight bits.
Less probability for corruption	Probability for corruption
Text file formats may include only text data.	Binary file formats may include multiple types of data in the same file, such as image, video, and audio data
Heavy weight in size.	Light weight in size





4.4 Input/output operations on text file

Character(text) files stores every type of data in the form of character only. C provides functions to perform input/output operations on text file such as `putc()`, `putw()`, `getc()`, `getw()`, `fprintf()`, `fscanf()`. These all functions are defined in `stdio.h` header file.

4.4.1 `putc()`

This library function is used to write single character to the specified stream.

Syntax#	Example#
<code>int putc(char c, FILE *fp);</code>	<code>putc('c',fp);</code>

It contains two arguments; first specify character `c` to be written in the file pointed to by `fp`. Second specify the file pointer associated with file.

If file **doesn't** exist it automatically creates new file.

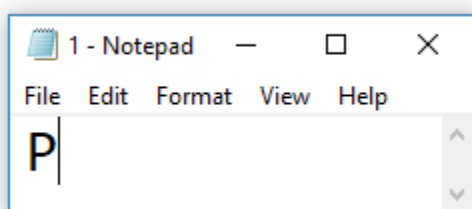


```
#include<conio.h>
#include<stdio.h>

main()
{
FILE *f1;
clrscr();

    f1=fopen("1.txt","w");
    putc('a',f1);
    fclose(f1);
    getch();
}
```

Output#



Steps to write the data in the file.

1. Declare a file pointer.
2. Open a file in write mode.
3. Reading data from the user.
4. Writing the data to the file.
5. Closing a file.

4.4.2 EOF (end of file)





Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

When reading or writing a file, to detect that the file pointer is reached at the end of the file, EOF is used.

In DOS, end of the file is indicated by a special character ctrl+z, in UNIX it is by ctrl+d.

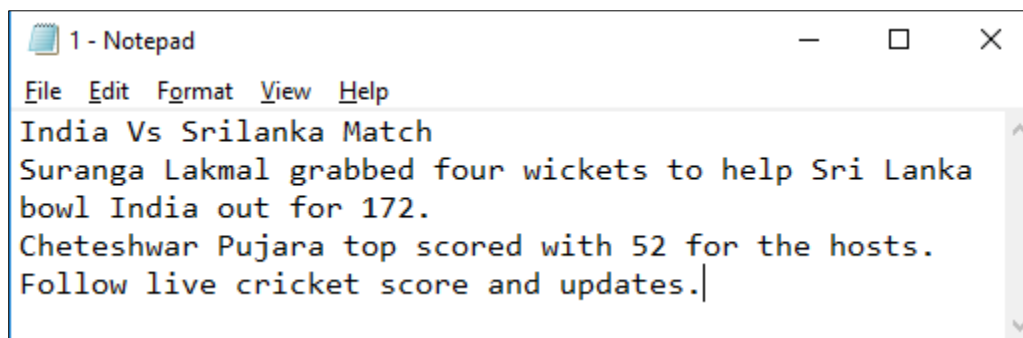
4.4.3getc()

It is used to input a single character from the file.



Syntax#	Example#
int getc(FILE *fp);	getc(fp);

Example# We have 1.txt file which contains some contents and now we are trying to read it



```
#include<stdio.h>
#include<conio.h>

main()
{
FILE *f1;
char ch;
clrscr();
f1=fopen("1.txt","r");
while(ch!=EOF)
{
    ch=getc(f1);
    printf("%c",ch);
}
fclose(f1);
getch();
}
```

Output#

```
India Vs Srilanka Match
Suranga Lakmal grabbed four wickets to help Sri Lanka bowl India out for 172.
Cheteshwar Pujara top scored with 52 for the hosts.
Follow live cricket score and updates.
```





Steps to read the data from the file.

1. Declare a file pointer.
2. Open a file in read mode.
3. Reading data from the file which is already exist.
4. Displaying data to user.
5. Closing a file.

Example# Check whether file exist or not

```
#include<conio.h>
#include<stdio.h>

main()
{
FILE *f1;
clrscr();

    f1=fopen("11.txt","r");

    if(f1 == NULL)
    {
        puts("\nDoes not exist\n");
        exit();
    }
    else
    {
        puts("\nFile is exist");
    }

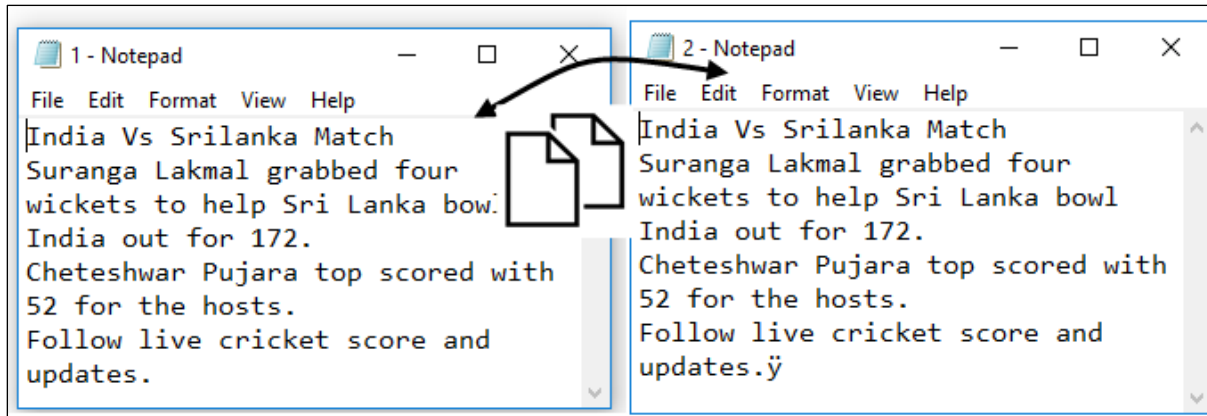
    fclose(f1);
    getch();
}
```

Output#

Does not exist

Example# Copy one file to another





```
#include<stdio.h>
#include<conio.h>

main()
{
FILE *f1,*f2;
char ch;
clrscr();

f1=fopen("1.txt","r");
f2=fopen("2.txt","w");

while(ch!=EOF)
{
    ch=getc(f1);
    putc(ch,f2);
}
fclose(f1);
fclose(f2);

    printf("\ncopied");

getch();
}
```

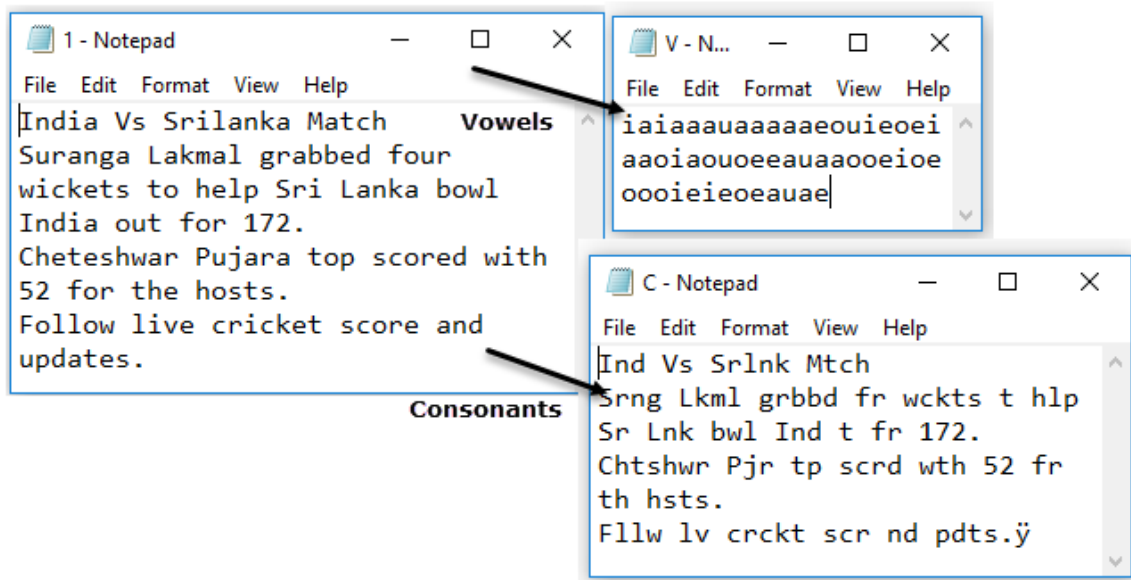
Output#

copied





Example# Read one file and divide vowels and consonants into other files



```
#include<stdio.h>
#include<conio.h>

main()
{
FILE *f1,*f2,*f3;
char ch;
clrscr();

f1=fopen("1.txt","r");
f2=fopen("v.txt","w");
f3=fopen("c.txt","w");

while(ch!=EOF)
{
ch=getc(f1);
if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
{
putc(ch,f2);
}
else
{
putc(ch,f3);
}
}
fclose(f1);
fclose(f2);
fclose(f3);
```





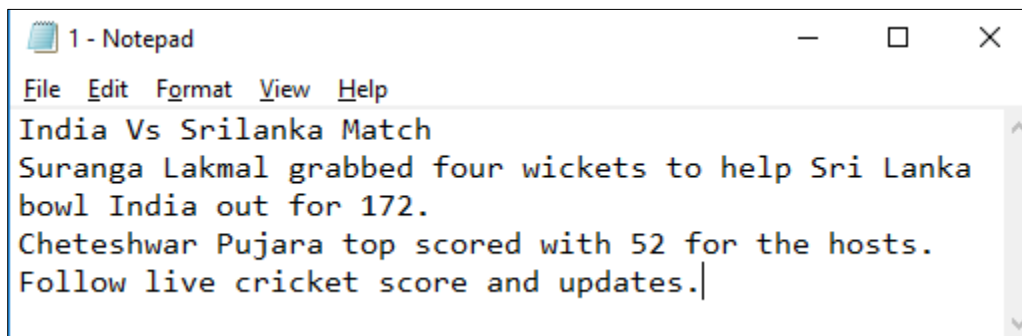
```
printf("\ncopied");
```

```
getch();  
}
```

Output#

copied

Example# Count no of spaces and lines



```
#include<stdio.h>  
#include<conio.h>  
  
main()  
{  
FILE *f1;  
char ch;  
int line=0,space=0;  
clrscr();  
f1=fopen("1.txt","r");  
  
while(ch!=EOF)  
{  
ch=getc(f1);  
  
    if(ch=='\n')  
    {  
        line++;  
    }  
    if(ch==' ')  
    {  
        space++;  
    }  
}  
fclose(f1);  
  
printf("\nNo of Lines = %d\nNo of Spaces = %d",line+1,space);
```





```
getch();
}
```

Output#

No of Lines = 4
No of Spaces = 31

4.4.4 Rewind()

The rewind() function is used to move the cursor **to** the beginning of the file.



Syntax#	Example#
void rewind(FILE *fp)	rewind(*fp)

Example# **Edit "1.txt" above file and append contents , rewind the cursor and print all the contents**

```
#include<stdio.h>
#include<conio.h>

void main()
{
    FILE *fp;
    char ch;
    clrscr();
    fp = fopen("1.txt","at+"); //append mode

    printf("\nWrite some data :\n");
    while((ch=getchar())!=EOF)
    {
        fputc(ch,fp);
    }

    rewind(fp); // after updating record cursor move to first

    printf("\nData in file :\n");
    while((ch = fgetc(fp))!=EOF)
        printf("%c",ch);

    fclose(fp);
    getch();
}
```

Output#





Write some data :

Do you know the latest score or any update about match?

Hey Tell me Karan.

Ctrl+Z

Data in file :

India Vs Srilanka Match

Suranga Lakmal grabbed four wickets to help Sri Lanka bowl India out for 172.

Cheteshwar Pujara top scored with 52 for the hosts.

Follow live cricket score and updates.**Do you know the latest score or any update about match?**

Hey Tell me Karan. //New contents added due to append mode

```

1 - Notepad
File Edit Format View Help
India Vs Srilanka Match
Suranga Lakmal grabbed four wickets to
help Sri Lanka bowl India out for 172.
Cheteshwar Pujara top scored with 52 for
the hosts.
Follow live cricket score and updates.Do
you know the latest score or any update
about match?
Hey Tell me Karan
    
```

4.4.5 getw()

This function is similar to getc() but it reads a integer value from the file.

The getw() function is used to read integer value form the file.

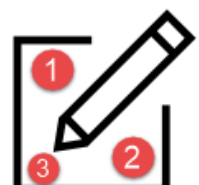
The getw() function takes the file pointer as argument from where the integer value will be read and returns the end-of-file if it has reached the end of file.



Syntax#	Example#
int getw(FILE *fp)	no=getw(f2)

4.4.6 putw()

This function is similar to putc() but it writes an integer value to the file.





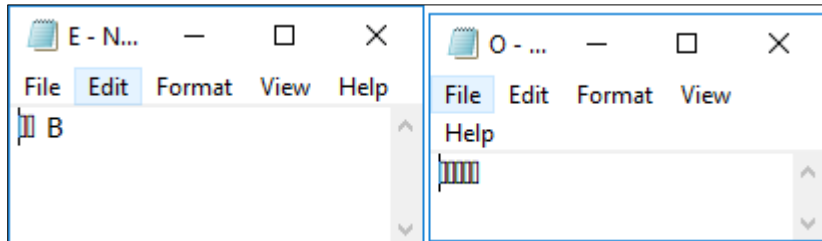
Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

The putw() function is used to write integers to the file.

The putw() function takes two arguments, first is an integer value to be written to the file and second is the file pointer where the number will be written.

Syntax	Example
putw(int number, FILE *fp);	putw(22,f1);

Example# Store 10 elements in one array, check the each value of array and if the value of array is even then copy it even.txt else odd.txt



```

#include<conio.h>
#include<stdio.h>

main()
{
    int ar[10]={1,5,3,4,6,7,11,2,66,9};
    FILE *f1,*f2;
    int i,no;
    clrscr();
        f1=fopen("o.txt","w");
        f2=fopen("e.txt","w");

        for(i=0;i<10;i++)
        {
            if(ar[i]%2==0)
                putw(ar[i],f2);
            else
                putw(ar[i],f1);
        }

        fclose(f1);
        fclose(f2);

        printf("\n\nOdd file \n\n");

        f1=fopen("o.txt","r");

        while((no=getw(f1))!=EOF)
        {
            printf("\n%d",no);
        }

```





```
fclose(f1);
printf("\n\nEven file \n\n");

f2=fopen("e.txt","r");

while((no=getw(f2))!=-1)
{
    printf("\n%d",no);
}
fclose(f2);
```

```
getch();
}
```

Output#

Odd file

1
5
3
7
11
9

Even file

4
6
2
66

4.4.7fscanf()

fscanf() function is used to read formatted data from a file.

This function is used to read different types of data from the file such as int, float, char etc.



Syntax#	Example#	
fscanf(fp, "control string", arguments list);	fscanf(f1,"%d %s",no,name);	

Control string is used to specify what types of data you want to read, Arguments list are the lists of variables separated by comma. fp is the file pointer points to the file which has been opened using fopen() function.





4.4.8 fprintf()

This function is used to write different types of data to the file such as int, float, char etc.

Syntax#	Example#
fprintf(fp, "control string", arguments list);	fprintf(f1,"%s %d",&no,&name);

Control string is used to specify what types of data you want to read, Arguments list are the lists of variables separated by comma. fp is the file pointer points to the file which has been opened using fopen() function.



Both functions fscanf() and fprintf() are used with binary data file.

Example# to read data from the keyboard and write to the file using fprintf() function

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int rollno,semester;
    char name[25], branch[25];
    int total;
    FILE *fp;
    char filename[15];
    clrscr();

    printf("Enter filename :");
    scanf("%s",&filename);

    fp=fopen(filename,"w");

    if(fp==NULL)
    {
        printf("\n file can not open for writing");
    }
    printf("\n Enter rollno, name , branch details:");
    scanf("%d %s %s",&rollno, &name,&branch);
    printf("\n Enter semester and total marks:");
    scanf("%d %d", &semester, &total);

    fprintf(fp,"%d %s %s %d %d", rollno, name, branch, semester, total);

    fclose(fp);
    getch();
}
```

Output#

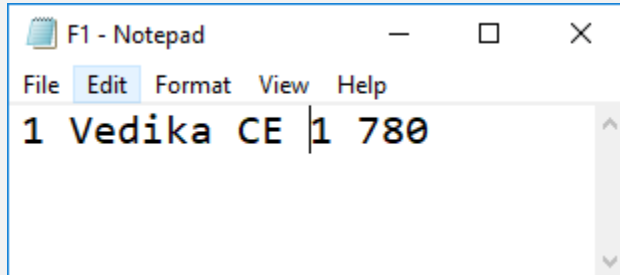




Enter filename :f1.txt

Enter rollno, name , branch details:1 Vedika CE

Enter semester and total marks:1 780



Example# After adding record in above program , to read data from the file and display to the file using fscanf() function

```
#include <stdio.h>
#include <conio.h>
void main()
{
    FILE *fp;
    int rollno, semester;
    char name[20], branch[20];
    clrscr();

    fp=fopen("f1.txt","r");

    if(fp == NULL)
    {
        printf("Cannot open file.\n");
        exit(1);
    }

    fscanf(fp,"%d%s%s%d", &rollno, name, branch, &semester);
    printf("\n%d %s %s %d", rollno, name, branch, semester);

    fclose(fp);
    getch();
}
```

Output#

1 Vedika CE 1





Example# fprintf() and fscanf()

```
#include <stdio.h>

#include <conio.h>
void main()
{
    FILE *fp;
    int rollno;
    char name[20];
    int i,n;
    clrscr();

    printf("\nEnter how many records u want to add =>");
    scanf("%d",&n);

    fp=fopen("f1.txt","w");

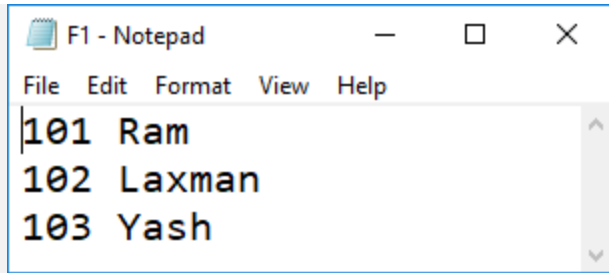
    for(i=0;i<n;i++)
    {
        printf("\nEnter no=>");
        scanf("%d",&rollno);
        fflush(stdin);
        printf("\nEnter name =>");
        gets(name);
fprintf(fp, "\n%d %s",rollno,name);
    }
    fclose(fp);

    fp=fopen("f1.txt","r");

    printf("\nRecords");
    for(i=0;i<n;i++)
    {
fscanf(fp,"%d %s", &rollno, name);
        printf("\n%d %s", rollno, name);
    }
    fclose(fp);
    getch();
}
```

Output#





```
F1 - Notepad
File Edit Format View Help
101 Ram
102 Laxman
103 Yash
```

Enter how many records u want to add =>3

Enter no=>101

Enter name =>Ram

Enter no=>102

Enter name =>Laxman

Enter no=>103

Enter name =>Yash

Records

101 Ram

102 Laxman

103 Yash





4.5 Input/output operations on binary file

When a large amount of numerical data is to be stored, text mode will be insufficient because memory occupation by it is large. In such case binary file is used. Working of binary files is similar to text files with few differences in opening modes, reading from file and writing to file. `fread()` and `fwrite()` functions are used with the binary file.

Opening a file

Binary file is open with same mode as with text file but character `b` is appended after the mode such as `wb`, `wb+`, `rb`, `rb+`, `ab`, `ab+` etc.

Reading and writing of a binary file

`fread()` function is used to read binary file and `fwrite()` function is used to write to the binary file.

4.5.1 fwrite()

The `fwrite()` function is used to write records (sequence of bytes) to the file. A record may be an array or a structure.

Syntax#	Example#
<code>fwrite(ptr, int size, int n, FILE *fp);</code>	<code>fwrite(&Stu1,sizeof(Stu1),1,fp);</code>

`fwrite()` function has four arguments:

- 1st argument indicate address of data to be written in disk,
- 2nd argument indicate size of data to be written in disk,
- 3rd argument indicate number of such type of data and
- 4th argument indicate pointer to the file where you want to write.



4.5.2 fread()

The `fread()` function is used to read bytes from the file.

Syntax#	Example#
<code>fread(ptr, int size, int n, FILE *fp);</code>	<code>fread(Stu1,sizeof(Stu1),1,fp);</code>

`fread()` function has similar four arguments as in `fwrite()`.

- 1st argument indicate address of data to be written in disk,
- 2nd argument indicate size of data to be written in disk,
- 3rd argument indicate number of such type of data and
- 4th argument indicate pointer to the file where you want to write.



4.5.3 fseek()

If we want to access the forty fourth record then first forty three records read sequentially to reach forty four records. In random access data can be accessed and processed directly.

There is no need to read each record sequentially. If we want to access a particular record random access takes less time than the sequential





access. C support fseek function for random access file.

fseek() function is used for setting the file position pointer at the specified bytes. fseek is a function belonging to the ANSI C Standard Library and included in the file stdio.h. Its purpose is to change the file position indicator for the specified stream.

Syntax#	Example#
int fseek(FILE *stream_pointer, long offset, int origin);	fseek(fp, 7, SEEK_SET)

stream_pointer is a pointer to the stream FILE structure of which the position indicator should be changed;

offset is a long integer which specifies the number of bytes from origin where the position indicator should be placed;

origin is an integer which specifies the origin position. It can be:

SEEK_SET	Origin is the start of the stream.
SEEK_CUR	Origin is the current position.
SEEK_END	Origin is the end of the stream.

Example# Structure Record Read write with Binary File

```
#include<stdio.h>
#include<conio.h>

struct Student
{
    int roll;
    char name[25];
    float marks;
};

void main()
{
    FILE *fp;
    char ch;
    struct Student Stu;
    clrscr();
    fp = fopen("StudentR.dat","w"); //Creates

    if(fp == NULL)
    {
        printf("\nCan't open file or file doesn't exist.");
        exit(0);
    }

    do
    {
        printf("\nEnter RollNo => ");
```





```
scanf("%d",&Stu.roll);

printf("Enter Name => ");
scanf("%s",Stu.name);

printf("Enter Marks => ");
scanf("%f",&Stu.marks);

fwrite(&Stu,sizeof(Stu),1,fp);

printf("\nDo you want to add another data (y/n) => ");
ch = getche();

}while(ch=='y' || ch=='Y');

printf("\nData written successfully...");

fclose(fp);

fp = fopen("StudentR.dat","r");//Reading from File

printf("\nRecords");
printf("\n\tRoll\tName\tMarks\n");
printf("\n=====");
while(fread(&Stu,sizeof(Stu),1,fp)>0)
{
    printf("\n\t%d\t%s\t%.2f",Stu.roll,Stu.name,Stu.marks);
}

printf("\n=====");
fclose(fp);
getch();
}
```

Output#

```
STUDENTR - Notepad
File Edit Format View Help
| Ram 1 85.00 @ 85.00
àA Mayur 2 90.00 @ 90.00
\B Vedika 3 78.00 @
Â! æB Mansi 4 88.00
@ Â! B Manav 5 92.00
@ Â! ,B
```





```
Enter RollNo => 1
Enter Name => Ram
Enter Marks =>28

Do you want to add another data (y/n) => y
Enter RollNo => 2
Enter Name => Mayur
Enter Marks =>55

Do you want to add another data (y/n) => y
Enter RollNo => 3
Enter Name =>Vedika
Enter Marks =>78

Do you want to add another data (y/n) => y
Enter RollNo =>4
Enter Name =>Mansi
Enter Marks =>39

Do you want to add another data (y/n) => y
Enter RollNo =>5
Enter Name =>Manav
Enter Marks =>65

Do you want to add another data (y/n) => n
Data written successfully...
```

Records

	Roll	Name	Marks
--	------	------	-------

=====		
1	Ram	28.00
2	Mayur	55.00
3	Vedika	78.00
4	Mansi	39.00
5	Manav	65.00
=====		

Example# Program to understand the use of fseek function , Read a given record of the student from the studentR.dat file.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
```

struct Student

```
{
    int roll;
```





```
char name[25];
float marks;
};
void main()
{
    int n;
    FILE *fp;
    struct Student s1;
    clrscr();

    fp=fopen("Studentr.dat","rb");

    if (fp==NULL)
    {
        printf("\n error in opening file");
        exit(1);
    }
    printf("\nEnter the record no to be read =>");
    scanf("%d",&n);

    fseek(fp,(n-1)*sizeof(s1),0);

    fread(&s1,sizeof(s1),1,fp);

    printf("\nRoll no = %d\nName = %s\nMarks = %.2f",s1.roll,s1.name,s1.marks);
    fclose(fp);
    getch();
};
```

Output#

Enter the record no to be read =>1

Roll no = 1

Name = Ram

Marks = 45.00

Example# Student Structure which contains roll,name,marks , write a contents in the file , read from the file , print only pass students, print specific records. Create function of each and create menu driven program.

```
#include<stdio.h>
#include<conio.h>

struct Student
{
    int roll;
    char name[25];
```



```
float marks;
};

void writeRecord()
{
    FILE *fp;
    char ch;
    struct Student Stu;

    fp = fopen("StudentRec.dat","w");

    if(fp == NULL)
    {
        printf("\nCan't open file or file doesn't exist.");
        exit(0);
    }

    do
    {
        printf("\nEnter RollNo => ");
        scanf("%d",&Stu.roll);

        printf("Enter Name => ");
        scanf("%s",Stu.name);

        printf("Enter Marks => ");
        scanf("%f",&Stu.marks);

        fwrite(&Stu,sizeof(Stu),1,fp);

        printf("\nDo you want to add another data (y/n) => ");
        ch = getche();

    }while(ch=='y' || ch=='Y');

    printf("\nData written successfully...");
    fclose(fp);
}

void printRecord()
{
    FILE *fp;
    struct Student Stu;

    fp = fopen("StudentRec.dat","r");

    printf("\nRecords");
    printf("\n\tRoll\tName\tMarks\n");
```





```
printf("\n=====");

while(fread(&Stu,sizeof(Stu),1,fp)>0)
{
    printf("\n\t%d\t%s\t%.2f",Stu.roll,Stu.name,Stu.marks);
}

    printf("\n=====");
    fclose(fp);
}

void printPassStudent()
{
    FILE *fp;
    struct Student Stu;

    fp = fopen("StudentRec.dat","r");

    printf("\nPass Students Records");
    printf("\n\tRoll\tName\tMarks\n");
    printf("\n=====");
    while(fread(&Stu,sizeof(Stu),1,fp)>0)
    {
        if(Stu.marks>50)
        {
            printf("\n\t%d\t%s\t%.2f",Stu.roll,Stu.name,Stu.marks);
        }
    }

    printf("\n=====");
    fclose(fp);
}

void printSpecificRec()
{
    FILE *fp;
    int n;
    struct Student Stu;
    fp=fopen("Studentr.dat","rb");

    if (fp==NULL)
    {
        printf("\n error in opening file");
        exit(1);
    }
    printf("\nEnter the record no to be read =>");
    scanf("%d",&n);
```





```
fseek(fp,(n-1)*sizeof(Stu),0);
fread(&Stu,sizeof(Stu),1,fp);
printf("\nRoll no = %d\nName = %s\nMarks = %.2f",Stu.roll,Stu.name,Stu.marks);
fclose(fp);

}

void main()
{
    char op;
    clrscr();

    writeRecord();

    do
    {
        printf("\nPress a for print all the Students records");
        printf("\nPress p for print only pass Students records");
        printf("\nPress s for print only specific position record");
        printf("\nPress e for exit =>");
        fflush(stdin);
        scanf("%c",&op);

        switch(op)
        {
            case 'a':
                printRecord();
                break;
            case 'p':
                printPassStudent();
                break;
            case 's':
                printSpecificRec();
                break;
            case 'e':
                exit(0);
            default:
                printf("\nWrong opt");
        }
        getch();
    }while(op!='e');

    getch();
}
```

Output#





```
STUDENTR - Notepad
File Edit Format View Help
Ram 1 28.00 @ A! 1 28.00
Mayur 2 55.00 @ A! 2 55.00
Vedika 3 78.00 @ A! 3 78.00
Mansi 4 39.00 @ A! 4 39.00
Manav 5 65.00 @ A! 5 65.00
```

Enter RollNo => 1

Enter Name => Ram

Enter Marks => 28

Do you want to add another data (y/n) => y

Enter RollNo => 2

Enter Name => Mayur

Enter Marks => 55

Do you want to add another data (y/n) => y

Enter RollNo => 3

Enter Name => Vedika

Enter Marks => 78

Do you want to add another data (y/n) => y

Enter RollNo => 4

Enter Name => Mansi

Enter Marks => 39

Do you want to add another data (y/n) => y

Enter RollNo => 5

Enter Name => Manav

Enter Marks => 65

Do you want to add another data (y/n) => n

Press a for print all the Students records

Press p for print only pass Students records

Press s for print only specific position record

Press e for exit

Press =>a

Records

	Roll	Name	Marks
=====			
1	Ram	28.00	
2	Mayur	55.00	
3	Vedika	78.00	





```
4  Mansi 39.00
5  Manav 65.00
```

=====

Press a for print all the Students records...menu
Press e for exit
Press =>p

Pass Students Records

Roll Name Marks

=====

```
2  Mayur 55.00
3  Vedika 78.00
5  Manav 65.00
```

=====

Press a for print all the Students records...menu
Press e for exit
Press =>s

Enter the record no to be read =>3

Roll no = 3
Name = Vedika
Marks = 78.00

Press a for print all the Students records
Press p for print only pass Students records
Press s for print only specific position record
Press e for exit
Press =>e

4.5.4 ftell()

ftell() in C is used to find out the position of file pointer in the file with respect to starting of the file.



Syntax#	Example#
long ftell(FILE *pointer)	int position; FILE *f1; f1=fopen("abc.txt","rb"); position=ftell(f1);

Example# Consider above Student file which contains three records and find total no of records and total no of bytes.





```
#include <stdio.h>
#include <conio.h>
```

```
struct Student
{
    int roll;
    char name[25];
    float marks;
};
```



```
main () {
    FILE *fp;
    int len;
    struct Student s1;
    int totalrec;
    clrscr();

    fp = fopen("StudentR.dat", "r");

    fseek(fp, 0, SEEK_END); // Pointer reach at the end of the file

    len = ftell(fp);

    totalrec=len/sizeof(s1);

    fclose(fp);

    printf("Total size of StudentR.dat file = %d bytes\n", len);
    printf("\nTotal Records are = %d",totalrec);

    getch();
}
```

Output#

Total size of StudentR.dat file = 155 bytes

Total Records are = 5





4.6 Command Line Arguments

It's the arguments supplied to main() program when the program is invoked. In common case this parameter or argument is a name of the file which main function should process.

Program execution always start with main() function. main() function can have arguments. After successful compilation and linking of the object file, compiler generates the executable file.

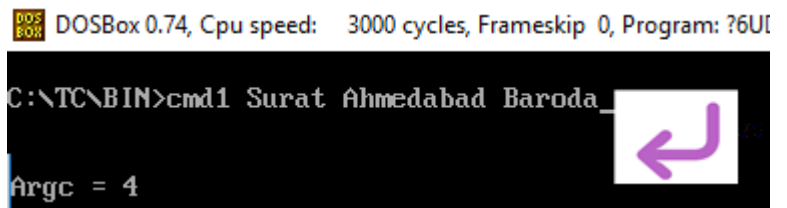
These arguments are passed with executable file. They are known as command line arguments because they are passed at the time of executing the program.

Format of the main() function with arguments :

```
void main(int argc, char *argv[])
```

int argc - presents total number of arguments pass by user

Name of the program itself argv[0] argument 0 so, argc always return 1 for it.



char *argv[] - presents the actual value of arguments

argv[0] reserved for program name

Whatever value user passes it will starts from argv[1]

Example#

If the executable file name is test.exe than in command line we can pass the arguments as:

```
c:\tc\bin>ProgramName Argument1 Argument2 Argument3
c:\tc\bin>test Prem Ratan Dhan
```

argc argument will stores the value 4 because there are three arguments passed including file name itself.

argv will stores the actual arguments as pointer to string.

argv[0] will stores test, argv[1] will stores Prem, argv[2] will stores Ratan, argv[3] will stores Dhan.

How To Run?

1. Press f9 key
2. Exe file is created for our program in TC\Bin (Check)
3. Clear the Run -> Arguments -> Clear it
4. File -> Dos Shell -> Go where your TC\Bin (Check)





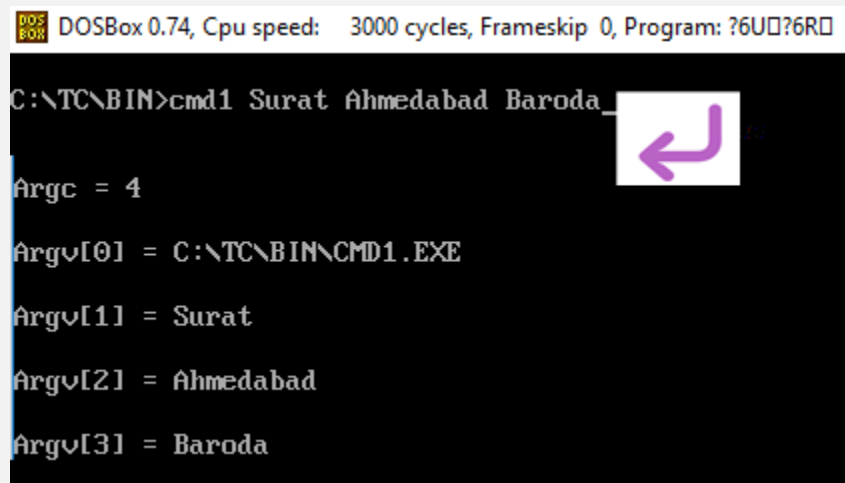
Example# Enter few values and display on command prompt

```
#include<conio.h>
#include<stdio.h>

void main(int argc,char *argv[])
{
    clrscr();
    printf("\n\nArgc = %d",argc);
    printf("\n\nArgv[0] = %s",argv[0]);
    printf("\n\nArgv[1] = %s",argv[1]);
    printf("\n\nArgv[2] = %s",argv[2]);
    printf("\n\nArgv[3] = %s",argv[3]);
    getch();
}
```

Output#

How to run? Press f9 then Go to Dos prompt and



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ?6U?6R

C:\TC\BIN>cmd1 Surat Ahmedabad Baroda_ ↵

Argc = 4

Argv[0] = C:\TC\BIN\CMD1.EXE

Argv[1] = Surat

Argv[2] = Ahmedabad

Argv[3] = Baroda

Example# Pass number of arguments on command line according arguments print all the values

```
#include<conio.h>
#include<stdio.h>

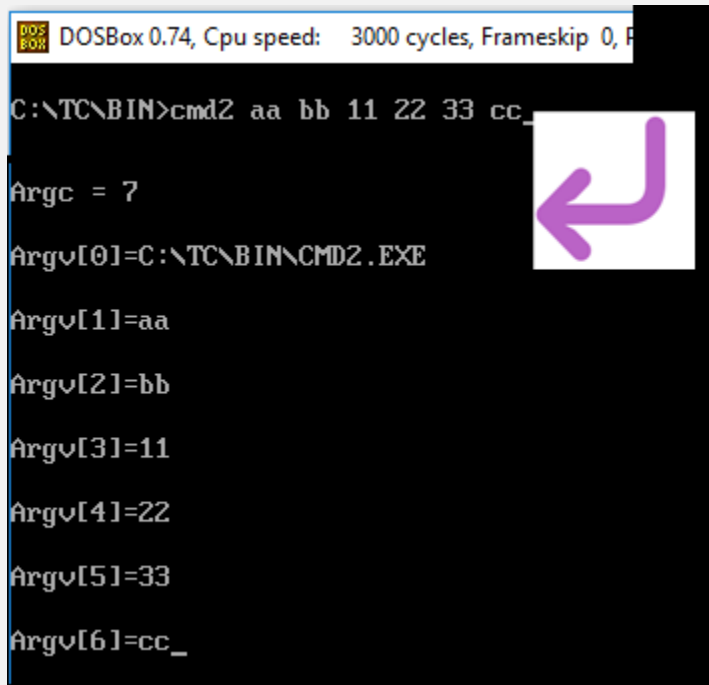
main(int argc,char *argv[])
{
    int i;
    clrscr();
    printf("\n\nArgc = %d",argc);
```





```
    for(i=0;i<argc;i++)
    {
        printf("\n\nArgv[%d]=%s",i,argv[i]);
    }
    getch();
}
```

Output#



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, P
C:\TC\BIN>cmd2 aa bb 11 22 33 cc_
Argc = 7
Argv[0]=C:\TC\BIN\CMD2.EXE
Argv[1]=aa
Argv[2]=bb
Argv[3]=11
Argv[4]=22
Argv[5]=33
Argv[6]=cc_
```

Example# Enter file name and print details

```
#include<conio.h>
#include<stdio.h>

main(int argc,char *argv[])
{
    char ch;
    FILE *read;
    clrscr();
    if(argc==2)
    {
        read = fopen( argv[1], "r" );

        while(ch!=EOF)
        {
            ch=getc(read);
```

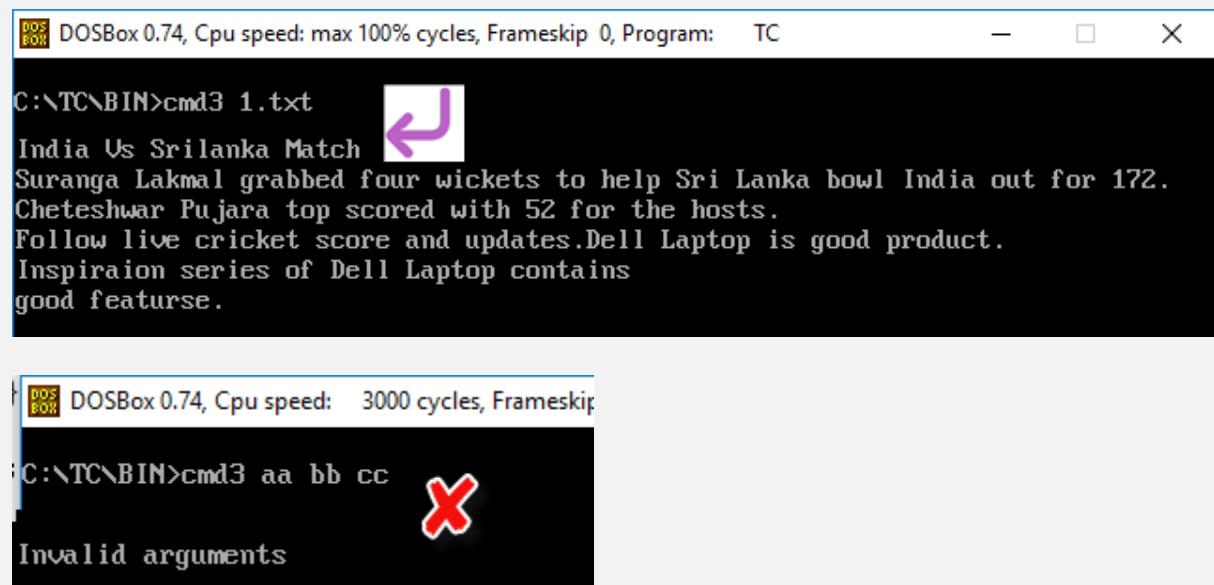




```
        printf("%c",ch);
    }
    else
    {
        printf("\nInvalid arguments");
    }

getch();
}
```

Output#



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

C:\TC\BIN>cmd3 1.txt
India Vs Srilanka Match
Suranga Lakmal grabbed four wickets to help Sri Lanka bowl India out for 172.
Cheteshwar Pujara top scored with 52 for the hosts.
Follow live cricket score and updates.Dell Laptop is good product.
Inspiraion series of Dell Laptop contains
good featurese.

C:\TC\BIN>cmd3 aa bb cc
Invalid arguments
```

Example# Merging of two file using command line

```
#include<conio.h>
#include<stdio.h>

main(int argc,char *argv[])
{
    char ch,ch1;
    FILE *f1,*af1,*f2;
    clrscr();
    if(argc==4)
    {
        f1=fopen(argv[1],"r");
        af1=fopen(argv[3],"w");

        while(ch!=EOF)
        {
            ch=getc(f1);
```





```
        putc(ch, afile);
    }

    fclose(f1);

    f2 = fopen(argv[2], "r");

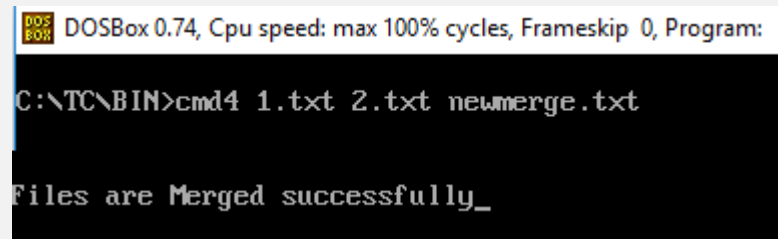
    while(ch1 != EOF)
    {
        ch1 = getc(f2);
        putc(ch1, afile);
    }
    fclose(f2);
    fclose(afile);

    printf("\nFiles are Merged successfully");

}
else
{
    printf("\nInvalid arguments");
}

getch();
}
```

Output#

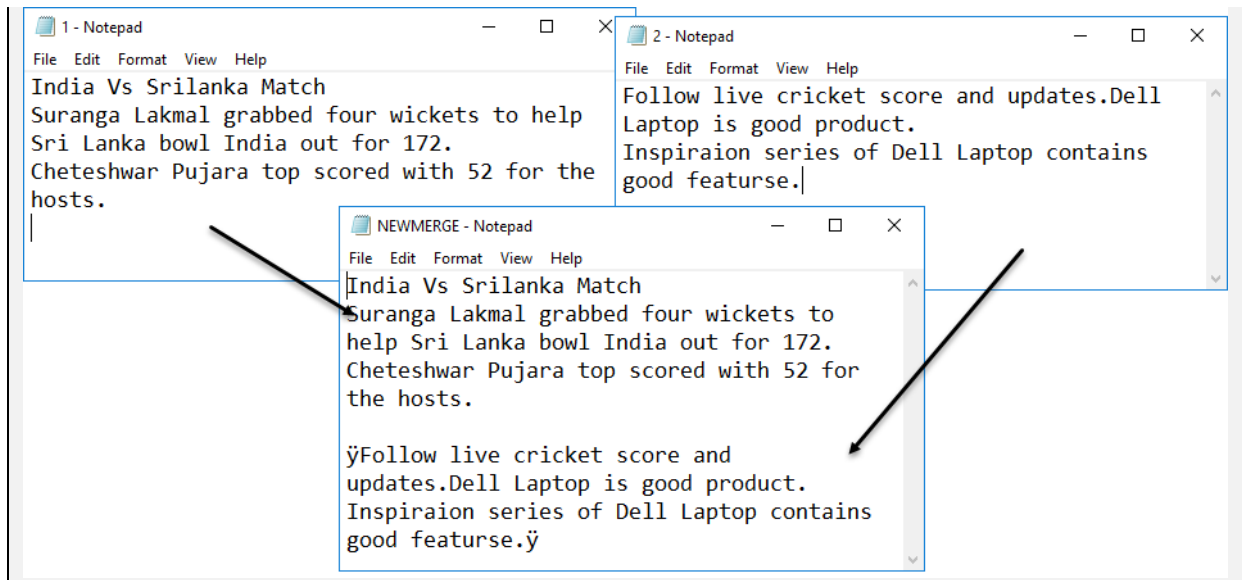


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:

```
C:\TC\BIN>cmd4 1.txt 2.txt newmerge.txt

Files are Merged successfully_
```







Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

