Till now we have used

- ✓ scanf() function to read data from <u>keyboard</u>
- ✓ printf() to write data on the <u>screen</u> (VDU).

This approach works fine for small data, BUT not suitable for large data volume because –

1. It becomes cumbersome & time consuming
2. The entire data is lost when program terminates / computer is turned OFF.

Hence we use <u>concept of files</u> to store data permanently so that we can RETRIEVE data as and when needed.

Informally file is a collection of bytes stored on secondary storage like Hard disk.

Basic operations on files include

- ✓ Creation of a NEW file
- ✓ Opening of an existing file
- ✓ Reading from a file
- ✓ Writing to a file
- ✓ Closing a file

  etc.

C provides library functions to perform above operations.

Let us NOW start formal discussion on file.

What is file?

A **file** is a named collection of data placed either in hard disks or in auxiliary storage devices. It can be accessed using the library functions or by the system calls of the operating systems.

What are the advantages of using file?

1. It allows us to store data permanently.
2. It allows us to access and modify data as and when required.

Types of data files in C:

1. **Stream-oriented data files/High-level data files** — these files are accessed by using library functions. It may of two types : -
   a) **Formatted** – data files are organized as sequence of characters. These characters can be interpreted as individual data items.

   b) Unformatted — this type of data file organizes data into blocks containing contiguous bytes of information. These blocks represent more complex data structure such as arrays and structures.

2. System-oriented data files/Low-level data files — these files are accessed using system calls of the operating systems. For example, read ( ), write ( ).

What is stream?

Stream is a source or destination of data that may be associated with a disk or other peripherals (Keyboard, VDU etc).

Stream may be of two types —

1. **Text Stream** – a text stream is a sequence of lines, each line has zero or more characters and is terminated by '\n'.
2. **Binary Stream** — it is a sequence of unprocessed bytes that records internal data with the property that if it is written, then read back on same system; it will compare equal.

REMEMBER –

A stream is connected to a file by opening it and the connection is broken by closing the stream.

File Operations –

Different operations on a file are as follows:

1. Creation of a new file
2. Opening an existing file
3. Reading from a file.
4. Writing to a file.
5. Moving to a specific location in a file.
6. Closing a file.

What is **FILE**?

FILE is a predefined structure data type that is defined in **stdio.h** header file.

NOTE that **FILE** and **file** are not same in this chapter. WHY?

What is **file pointer** (stream pointer)?

A file pointer is a pointer that points to a structure of type **FILE.** The structure **FILE** contains information about the file such as location of the buffer, current character position in the buffer, whether file is being read or written etc.

Example:

FILE * fp ;

Here, fp is a file pointer that points a structure of type FILE.

WHY & HOW to open a file:

In order to use a file on disk we need to establish a connection with it. A connection is established using the **fopen** function. The general form is

fp = fopen(fname , mode) ;

where **fname** represents name of the file to be accessed
mode denotes the purpose of opening the file.
fp is the file pointer.

.

File Modes:
File mode indicates how one intends to use the file. That is whether the file is open for reading, writing or appending data.

1. "r" — Opens an existing text file for reading only. If the file does not exist then error occurs.
2. "w" — If specified text file does not exist, a new file is created for writing only otherwise content of the existing file is deleted.
3. "a" — If the specified text file does not exist, a new file is created otherwise open the existing text file for appending.(i.e. writing at end of file)
4. "r+" — Open the existing text file for both reading and writing.
5. "w+" — create text file for reading and writing, discards previous contents if any.
6. "a+" — Open or create text file for both reading and appending.
7. "rb" – open binary file for reading
8. "wb" – create binary file for writing, discard previous contents if any.
9. "ab" – open or create a binary file for appending

Review Question :
What are the meaning of the following modes –
"r+b", **"w+b"**, **"a+b"**

Hint: "r+b" — Open the existing binary file for both reading and writing.

How to read or write the file once it is open:

getc and putc are simplest file I/O functions to handle one character at a time.

getc():

It is used to read a character at a time from a file that is open in a read mode.

Example:

c = getc(fp);

This statement would read a character from a file whose file pointer is **fp**. It returns the character read from the stream referred to by "fp".

The function fgetc() is equivalent to getc() and can be used interchangeably.

putc():

It is used to write a character onto a file that is open in write mode.

Example:

putc(c, fp);

This statement writes the character contained in the character variable **"c"** to the file associated with file pointer **fp**.

The function fputc() is equivalent to putc() and can be used interchangeably.

---

getw is used to read integer values from a file. putw is used to write integer values to a file that is open in write/append mode.

How to close a file:

The function **fclose** is used for closing a file. The general form

fclose(fp);

where fp is the file pointer of the file that one intends to close.

What do you mean by stdin, stdout and stderr?

When a C program begins execution, OS environment opens three files – the standard input, the standard output and the standard error files. The corresponding file pointers are called stdin, stdout and stderr. They are declared in stdio.h header file. Normally **stdin** is connected to the keyboard and **stdout** and **stderr** are connected to the screen.

--

What is EOF and how does it help to detect the end of a file?

EOF, a symbolic constant, is defined in stdio.h as –1 , a value never used by real character.
Using EOF we can detect the end of a file. We will take an example to explain.

```
        while( (c = getc(fp) ) != EOF)
```

Here, getc() is used to read a character from a file whose file pointer is fp and that has been opened in read mode. The file pointer moves by one character position for every operation of getc(). The getc() will return an end-of-file marker EOF, when end of the file is reached. So, the reading should be terminated when EOF is encountered,

----

Remember ctrl-z is used as end-of-file character in DOS. ctrl-d is used as end-of-file character in Unix/Linux.

Prog 1: To display content of an existing file.

```
#include <stdio.h>
int main()
{
        FILE *fp;
        char c,  str[50];
        printf("\nEnter the name of the file to be read : ");
        gets(str);
        fp=fopen(str, "r");
        printf("\nContent of the given file\n\n");
        while( (c = getc(fp) ) != EOF)
        {

                printf("%c", c);
        }
        fclose(fp);
        return(0);
}
```

Prog 2: To take input a text from keyboard and write onto a file. Also display the content of file in the screen.

```
#include <stdio.h>
int main()
{
        char c;
        FILE *fp;
        fp=fopen("x.dat","w");
        printf("\n Input data\n");
        while((c=getchar())!=EOF)
        {
                 putc(c,fp);
        }
        fclose(fp);
        fp=fopen("x.dat","r");
        printf("\n Contents of the file\n");
        while((c=getc(fp))!=EOF)
        {
                printf("%c",c);
        }
        fclose(fp);
        return(0);
}
```

Prog 3: WACP to copy one disk file into another disk file using command line arguments.

```
#include <stdio.h>
int main(int argc, char* argv[])
{
        FILE *fs,*fd;
        char c;
        fs=fopen(argv[1],"r");
        fd=fopen(argv[2],"w");
        while((c = getc(fs)) != EOF)
                putc(c,fd);
        fclose(fs);
        fclose(fd);
        return(0);
}
```

Prog 4: Write a C program to display the frequency of characters in a given disk file.

```
    #include <stdio.h>
    int main()
    {
                FILE  *fs;
                int  i;
                int  n[256]={0};
                char  c;
                fs = fopen( "x.txt", "r");
                while( (c = getc(fs) )  !=  EOF)
                        n[c] = n[c] + 1;
                fclose(fs);
                for(i = 0; i <= 255; i++)
                        {
                        if( n[i] != 0)
                        {
                                if(i==10)
                                        printf("Frequency of  \\n  character is  %d \n", n[i] );
                        else
                                    printf("Frequency of  %c  character is %d \n", i , n[i] );
                        }
                }
                return(0);
      }
```

Prog 5: Write a C program to find no. of characters, words and lines.

```
#include <stdio.h>

int  main()
{
        FILE *fs;
        int lc, wc, cc;
        char c;
        clrscr();
        lc=wc=cc=0;
        fs=fopen("x.dat","r");

        while((c=getc(fs))!=EOF)
        {
                if(c>=65 && c<=122)
                        cc++;

                if(c==' ' || c=='\n' || c=='\t')
                        wc++;

                if(c=='\n')
                        lc++;
        }
        fclose(fs);

        printf("\n No. of characters is %d",cc);
        printf("\n No. of words      is %d",wc);
        printf("\n No. of lines      is %d",lc);
        return(0);
}
```