Date 18 march 20
Page

UNIT NO. File & File Handling

File :- A file is collection of related data placed on a disk.

After determination of any program all entered Data is lost because primary memory is Volatile. If we want to use the data later it is necessary to keep save it in a permanent storage device. 'C' Support the concept of file through which data can on disk OR Secondary Storage. This concept of storage data is fulfill with the help of files.

File Handling in C can be divided into 2 Category :-

High Level
Low Level

High Level or Standard file or Stream Oriented file :-

High Level file Handling is manage by library function Ex- different function like printf, Scanf is Handle by Stdio, Conio.

Low Level or System Oriented Files :-

Low level file Handling is manage by System Call.

Types of files — Files are Two Types.

Scanned with CamScanner

(1). Text File
(2). Binary file

Text File :- In text file data is stored as line of character with each line terminated by a new line character. In text file same conversation have to take placed between memory & file.
Ex- notepad.

Binary File :- In binary file data is stored on the disk in the same way as stored in computer memory & in binary file their are No- conservation needed between memory & file.

Buffer :- Buffer is an area in memory where data is stored temprory before written to the file. where we open a file. A buffer is automatically with its file pointer, whatever data we send to the file is not immediately return to the file. first it is return is buffer & when buffer is full. It is return to the file & when file is closed the remaining data of buffer is automatically inserted to the file. The concept of buffer increase the efficiency.

Different operation Related :-
(1). Open a file or create a file
(2). Read or write in the file.
(3). Close the file.

Scanned with CamScanner

(1) Open a file or create a file :- A file must be open before any I/O operation perform on the file. The process of establishing a connection between the program & a file is called opening a file. A structure name (file) is defined in the <stdio.h> & all the information related to file like file names, file status, file size, current positions the file & end of the file status are stored. Here.

```
type def struct
{

} FILE
```

A file pointer is pointer to a structure of type file, whenever a file is created a structure of type FILE is associated with it & a file pointer that point to this structure identify this file. Here we use the function fopen to create a file.

Declaration :-

```
File *fopen (const char * file name, const char*
                                        )

File * fp
fp = fopen ("My text.txt", "is");

type def struct
{

} FILE
```

Scanned with CamScanner

Date 18 March 20
Page

* fOpen function takes two String (Value) as argument the first one is that name of the file & Second one is the mode that decide which operation perform. On Success, fopen return a pointer of type file & if error than return

Scanned with CamScanner

Structure of General File Program :-

```
main ( )
{
 file * fp
 fp = fopen ("file", "mode");
 =
 =
 fclose (fp);
}
```

There are two argument in opening a file first argument define the name of the file & Second argument define the mode, mode define the type of operation perform in a file. They are different type.

i). WRITE ("W")
ii). READ ("R")
iii). APPEND ("A")
iv). WRITE AND READ ("W+")
v). READ AND WRITE ("R+")
vi). APPEND + READ ("A+")

(i). WRITE ("W") :- If the file does not exist then this mode create a new file & if the file already exist then previous Data erase & new data Create.

(ii). APPEND ("A") :- If the file doesnt exist this mode create a file and

Scanned with CamScanner

if the file is ~~always~~ already exist then this mode attach the new data below the previous data.

(iii). READ ("R"):- This mode is only used for Reading Purpose.

(iv). READ-WRITE ("R+"):- This mode is same as R-Mode but in this mode we can also write and modify Data.

(v). WRITE - READ ("W+"):- This mode is same as W-mode but It also include Read or Modify the Data.

(vi). APPEND + READ ("A+"):- This Mode is similar to A-Mode but it also include the Read and modify data.

Closing A file:-

```
int fclose (FILE *fp);
```

END OF FILE:- The file reading function needs to know the end of the file so that they can stop reading for this purpose O.S, send an END of the file signal to the program & when the program receive this signal, the file reading function return which is a constant whose value is define in stdio.h and its value is (-1).

Scanned with CamScanner

M. Imp

* **Closing a file :-** The file that was open using the (Fopen) function must be closed when No-more operations are to be performmed on it. After Using the file the collection between the file and the program is closed.

Declaration ⇒ `int fclose (FILE *fptr);`

* **Pre-Define file pointer :-** Three pre define constant file pointer are open automatically when the program is Executed three pointer are —

① Stdin

② Stdout

③ Std err

Function used in file :- There are different types of function which are used in file Handling.

i) Character file / output / input → fgetc ( ), fputc ( ), getc ( ), putc ( ).

Scanned with CamScanner

ii). String file Input/output ⇒ fgets(), fputs()

iii). Int file Input/output ⇒ getw(), putw().

iv). Formatted input/output ⇒ fread(), fwrite()

* Char input/output file :-

(i). fputc() :-

declaration ⇒ int fputc (int c, file *p);

This function write a character to the specified file at the current file position and then increment the file position pointer on access it return on (EOF) end of the file.

```
# include <Stdio.h>
# include <Conio.h>
Void main ()
{
FILE *fp;
int ch;
if ((fp = fopen ("my file.txt")) == NULL)
{
Printf ("File not exist");
exit ();
}
Else
{
Printf ("enter text");
```

Scanned with CamScanner

```
while (ch = getchar() != EOF)
    fputc(ch, fp);
}
fclose(fp);
}
```

(ii) **Fgetc ( ) :-** This function read a Single Character from a given file and increment the file pointer position on success, it return the Character after converting it to an integer without sign extension, on the End of the file or error it Return (EOF).

declaration ⇒ int fgetc (FILE *fp);

```
# include <Stdio.h>
# include <Conio.h>
    Void main ( )
  {
   FILE *fp;
   int ch;
   if ((fp = fopen ("my file.txt")) == NULL)
  {
   Printf ("File not exist");
   exit ();
   }
   else
   {
   while ((ch = fgetc (fp)) != EOF)
    printf ("%c", ch);
   }
```

Scanned with CamScanner

fclose (fp);
}

* **Random function ( ) :-** In file we Access the data by two way-

1). Sequential
2). Random

There are three function which are Used to Access the data randomly.

(i). f Seek ( )
(ii). f tell ( )
(iii). rewind ( )
} Random function

(i). **f Seek ( ) :-** This function is used for setting the file position, pointer at the Specified Byte.

declaration => int fseek (FILE *fp long displacement, int origin);

Here fp is file pointer
Displacement → long integer (+ or −)
it denotes the number of bytes which are Skipped backward (if −ve) & forward (if +ve) from the position specified in the third argument. Name origin is the position related to which displacement take place, it can take three Value.

1). Zero → It indicate the degree beginning of the file and it is represented a

Constant Seek Set and

```
# include <stdio.h>
# include <conio.h>
  Void main ()
  {
   FILE *fp;
   fp= fopen ("data.txt", "r");

   if (fp == NULL) {
    printf ("FILE does not exist, please
                                check !\n");
   }
    fclose (fp);
  }
```

Imp **fopen() :-** This function accepts two argument as strings. The first argument denotes the name of the file to be opened and the second signifies the mode in which the file is to be opened. The second argument can be any of the following.

File Mode	Description
r	Open a text file for reading
w	Create a text file for writing, if it exist, it is over written.

Scanned with CamScanner

a	Open a text file and append text to the end of the file.
rb	Open a binary file for reading.
wb	Create a binary file for writing. if it exist, it is overwritten.
ab	Open a binary file and append data to the end of the file.

fclose() :- The fclose() function is used for closing opened files. The Only argument it accepts is the filepointer.

If a program terminates, it automatically closes all opened files. But it is a good Programming habit to close any file once it is no longer needed. This helps in better Utilization of System resources & It is very useful when you are working on numerous files Simultaneously. Some Operating System place a limit on the number of the files that can be open at any given point in time.

Imp.

fScanf() and fprintf() :-

The function fprintf() & fScanf are similar to printf() & Scanf() except that these function operates on file and

require one additional and first argument to be a file pointer.

```
# include <Stdio.h>
# include <Conio.h>
    Void main ()
    {
  FILE *fp;
  float total;
  fp = Fopen ("data.txt", "w+");
  if (fp == NULL) {

Printf ("data.txt does not exist, Please check!\n");
  exit (1);
}
  fprintf (fp, 100);
  fScanf (fp, "%f", & total);

  fClose (fp);
  printf ("Value of total is %f\n", total);

}
```

Scanned with CamScanner

Programs examples

File : Ftell.c

```
#include<stdio.h>
# include<conio.h>
  Void main()
  {
    FILE *fp;
    int length;
    clrscr();
    fp = fopen("File.txt","r");
    f seek (fp, 0, SEEK_END);

    length = ftell(fp);


    fclose(fp);
    printf ("size of file : %d bytes", length);
    getch();
  }
```

Output :- size of file : 21 bytes

M. Imp

C-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.

Scanned with CamScanner

Rewind function (Rewind File) :- {Random function}

In the C Programming Language, the rewind function sets the file position to the beginning of the file for the stream pointed to by stream.

It also clears the errors and end-of-file indicators for stream.

Syntax :- Void rewind (FILE * stream);

C-Library function - feof( ) :-

The C library function int feof (FILE * stream) tests the end-of-file indicator for the given stream.

Declaration of feof( ) -

```
int feof(FILE * stream)
```

Parameter -

Stream - This is the pointer to a FILE object that identifies the stream.

Return Value. -

This function returns a Non-zero value when End-of-file indicator associated with the stream is set else zero is Returned.

Scanned with CamScanner

Example -

```
#include <Stdio.h>
#include <Conio.h>
Void main ()
{
  int c;
  FILE * fp;

  fp = fopen ("File.txt", "r");
  if (fp == NULL)
  {
    Perror ("Error in Opening File")
    return (-1);
  }
  while (1)
  {
    c = fgetc (fp);
    if ( feof (fp))
    {
      break;
    }
    Printf ("%c", c);
  }
  Fclose (fp);
  return (0);

}
```

Scanned with CamScanner

Date 6 April

Bitwise Operators :- Bitwise operators are used for manipulating a data at the bit level. also called as bit level programming. Bit-level programming mainly consist of 0 & 1. They are used in numerical computations to make the calculation process faster.

C provides six operators for bit manipulation.

Symbol	Operator
&	bitwise AND
|	bitwise inclusive OR
^	bitwise XOR (exclusive OR)
<<	left shift
>>	right shift
~	bitwise NOT (one's complement) (Unary)

Scanned with CamScanner

Date 7 April

Imp

Bitwise AND :-

This is One of the most Commonly Used logical bitwise Operators. It is represented by a Single ampersand Sign (&). Two integer expression are written on each side of the (&) Operator.

The Result of the bitwise AND operation is 1 if both the bits have the value as 1; otherwise, the result is always 0.

Let us Consider that we have 2 variables op1 & op2 with Values as follows:-

```
  op1 = 0000 1101
& op2 = 0001 1001
```

The result of the AND operation on variables op1 & op2 will be

Result = 0000 1001

As we can see, two Variables are Compared bit by bit.

Whenever the Value of a bit in both the Variables is 1, then the result will be 1 or else 0.

multiplication

Date 8 April

Bitwise OR :-

It is represented by a single vertical bar sign (|). Two integer expressions are written on each side of the (|) operator.

The result of the bitwise OR operation is 1 if at least one of the expression has the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 & op2 with values as follows:

op1 = 0000 1101
op2 = 0001 1001

The result of the OR operation on variables op1 & op2 will be

$$\boxed{\text{Result} = 0001\ 1101}$$

As we can see, two variables are compared bit by bit. Whenever the value of a bit in one of the variables is 1, then the result will be 1 or else 0.

Addition

Date 9 April 20
Page

Bitwise XOR (exclusive OR) operator ^ :-

It is represented by a symbol (^). Two integer expressions are written on each side of the (^) operator.

The result of the bitwise Exclusive-OR operation is 1 if only of the expression has value as 1; otherwise, the result is always 0.

Let us consider that we have 2 Variables op 1 & op 2 with Values as follows:-

```
  op 1 = 0000 1101
^ op 2 = 0001 1001
```

The result of the OR operation on Variables op1 & op2 will be

$$\boxed{\text{Result} = 0001\ 0100}$$

As we can see, two Variables are compared bit by bit. Whenever only one variable holds the value 1 then the result is 0 else 0 will be the result.

Input & Opposite result

eg.

```
  0     0     1
  1     0     1
 ---   ---   ---
  1     1     0
```

Scanned with CamScanner

Date 10 April 20

Bitwise Complement Operator ~ :-

Bitwise Compliment operator is an unary operator (work on only one operand.). It changes 1 to 0 & 0 to 1. It is denoted by ~.

35 = 00100011 (In Binary)

Bitwise complement operation of 35

~ 00100011

11011100 = 220 (In decimal)

Shift Operators in 'C' Programming

There are two Shift operators in C Programming:-

- Right Shift operator
- Left Shift operator.

Right Shift Operator

Right Shift Operator Shift all bits towards right by certain number of specified bits. It is denoted by Sign >> .

212 = 11010100 (in binary)

212 >> 2 = 00110101 (in binary) [Right Shift by two bits]

212 >> 7 = 00000001 (in binary)

212 >> 8 = 00000000 (in

212 >> 0 = 11010100 (NO Shift)

Scanned with CamScanner

Date 11 April 20
Page

Left Shift Operator

Left Shift Operators shifts all bits towards left by certain Number of specified bits. It is denoted by sign <<.

212 = 11010100 (In binary)

212 << 1 = 110101000 (in binary) [Left shift by one bit]
212 << 0 = 11010100 (Shift by 0)
212 << 4 = 110101000000 (in binary) = 3392 (in decimal)

Scanned with CamScanner