FILES

<u>Definition:</u> A file is a collection of bytes stored on a secondary storage device, which is generally a disk of fame kind.

A file is a collection of related data stored on a hard disk or secondary storage device. Files can be accessed using library functions.

The files accessed though the library functions are called 'stream oriented files' and the files accessed with system calls are known as 'system oriented files'.

<u>Stream</u>:- stream means reading and writing of data. Stream is classified into 3 types and those are defined in io.h header file.

1.Stdin : reads input from the keyboard

2.stdout : send output to the screen.

3.stderr : print errors to an error device.

Based on internal storage representation files are two types:-

1. Text files
2. Binary files

<u>**Text File**:-</u> (COLLECTION OF CHARACTER STORES IN A FILE)

The I/O stream can be a text stream or binary stream. A text steam is a sequence of text. In a text stream, each line consists of zero or ore characters, terminated with newline character.

<u>**Binary File**:-</u> (collection of ASCII values stores in a file)

A binary stream is a sequential of bytes. In a binary stream, it represents raw data without any conventions.

Based on accessing files are two types:-

1. Sequential file.
2. Random access file.

<u>**Sequential File**</u>:- In this type of files records are kept sequential. If we want to read the last record of the file we need to read all the records before that record.

<u>**Random Access File**</u>:- In this type of files data can be read and modified randomly. In this type if we want to read the last records of the file, we can read it directly.

FILE: In C, FILE is a structure that holds the description of a file and is defined in stdio.h. It contains the current status of the file. FILE is a data type and a region of storage.

File pointer:

Declaration of file pointer is as follows,

*FILE *variable-name;*

Where variable-name refers to the name of the pointer variable.

Ex:-

```
FILE  *fp1,*fp2;
FILE  *fptr;
```

- A file pointer is a pointer to FILE data type.
- File pointer is also called as a stream pointer or buffer pointer.
- A file pointer points to the block of information of the stream that has just been opened.

File Operations:-

a) Creation of a new file.
b) Opening an existing file.
c) Reading from a file.
d) Writing to a file.
e) Moving to a specific location. (seeking)
f) Closing a file.

FILE FUNCTIONS:-

FUNCTION	**OPERATION**
fopen ( )	Creates a new file for read/write operations.
fclose( )	Closes a file associated with file pointer.
closeall( )	closes all opened files with fopen( )
fgetc( )	Reads the character from current pointer position. And advances the pointer to the next character.
getc( )	Reads the character from current pointer position. And advances the pointer to the next character.
fprintf( )	Writes all types of data values to the file.
fscanf( )	Reads all types of data values from a file.
putc( )	Writes character one by one to a file.
fputc( )	Writes character one by one to a file.
gets( )	Reads string from the file.
puts( )	writes a string to the file
putw()	Writes integer to the file.

getw( )	Reads an integer from the file.
fread( )	Reads structure data written by fwrite( ) function.
fread()	Reads structure data written by fwrite( ) function.
fwrite( )	Write block of structure data to the file.
fseek( )	Sets the pointer position anywhere in the file.
feof( )	Detects the end of the file.
ferror( )	Reports error occurred while read/write operations.
perror( )	Prints compiler errors messages along with user defined messages.
ftell( )	Returns the current pointer position.
rewind( )	Sets the file pointer at the beginning of the file.
unlink( )	Removes the specified file from the disk.
rename( )	Changes the name of the file.

FILE OPENING MODES:-

MODE	DESCRIPTION
"r"	Reading from the file.
	Searches the file if the file is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first character in it.
	If the file cannot be opened fopen( ) returns NULL.
"w"	Writing to the file.
	Searches the file, if the file exist its contents are overwritten.
	If the file doesn't exist a new file is created.
	If the file cannot be opened fopen( ) returns NULL.
"a"	Appends new content at the end of file.
	Searches the file if the file is opened successfully fopen( ) loads it into the memory and sets up a pointer that points to the last character in it.
	If the doesn't exist a new file is created.
	If the file cannot be opened fopen( ) returns NULL.

"r+"	Reading existing contents, creating new contents. modifying existing contents of the file. Searches file, if it is opened successfully. Fopen( ) loads it into memory and sets up a pointer which points to the first byte character in it. If the file cannot be opened fopen( ) returns NULL.
"w+"	Reading existing contents, writing new contents. modifying existing contents of the file. Searches file, if it is opened successfully its contents are overwritten. If the file doesn't exist a new file is created. If the file cannot be opened fopen( ) returns NULL.
"a+"	Reading existing contents, appending new contents to the end of the file. Cannot modify existing contents of the file. Searches file, if it is opened successfully fopen( ) loads it into memory and sets up a pointer which points to the first byte character in it. If the file doesn't exist a new file is created. If the file cannot be opened fopen( ) returns NULL.

FILE MODES TABLE:-

Mode	**r**	**w**	**a**	**r+**	**w+**	**a+**
Open state	Read	Write	Write	Read	Write	write
Write allowed	yes	no	no	yes	yes	yes
Append allowed	no	yes	yes	yes	yes	yes
File allowed	no	no	yes	no	no	yes
File must exist	no	yes	no	yes	no	no
Contents of existing file lost	no	yes	no	no	yes	no

Buffer:- a buffer is a temporary storage area that holds data while they are being transfered to or from memory.

Standard Library Input/ Output Functions:-

1. File open/close.
2. Formatted input/output.
3. Character input/output.

File Open/Close:-

fopen( ):-

fopen( ) performs three important tasks when you open the file in 'r' mode.

1. Firstly it searches on the hard disk the file to be opened.
2. Then it loads the file from the disk into a place in memory called buffer.
3. It sets up a pointer that points to the first character of the buffer.

Syntax:- *fopen("file name","mode");*

fclose( ):-

When we no longer need an opened file, we should close it to free system resources, such as buffer space. A file is closed using the close function fclose( ).

Syntax:

```
                fclose(file_ptr);
```

```
//Write a program to read the file and display its contents on the screen.
#include<stdio.h>
int main( )
{
 FILE *fp;
 char ch;
 fp=fopen("factorial.c","r");
 while(1)
  {
   ch=fgetc(fp);
   if(ch==EOF)
          break;
   else
   printf("%c",ch);
  }
 fclose(fp);
 getch( );
}
```

Files

```
//Write a program to open a file and display the contents of the file and close that file.
#include<stdio.h>
void main( )
{
  FILE *fp;
  char ch;
  clrscr( );
  fp=fopen("poly.c","r");
  ch=fgetc(fp);
  while(ch!=EOF)
  putchar(ch);
  fclose(fp);
  getch( );
}
```

fputc() and fgetc():

Syntax:

```
          fputc(int ch, file pointer);
          fgetc(file pointer);
```

```
//Write a program to write some characters on file and read them back.
#include<stdio.h>
main( )
{
     FILE *fptr;
     char ch;
     clrscr( );
     fptr=fopen("new.txt","w");
     printf("enter some characters on file");
     while((ch=getchar( ))!=EOF)
     fputc(ch,fptr);
     fclose(fptr);
     fptr=f open("new.txt""r");
```

```
        while((ch=getchar( ))!=EOF)
        ch=fgetc(fptr);
        fclose(fptr);
        getch( );
}
```

Formatted I/O:-

1. fprintf( ):-

   syntax:- printf("control string", variable-list);

   fprintf(file-pointer, "control string", variable list);

2. fscanf( ):-

   syntax:- scanf("control string",variable-list);

   fscanf(file-pointer, "control string", variable list);

```
//program that shows the example of fprintf( ) and fscanf( ).
#include<stdio.h>
#include<io.h>
#inlucde<conio.h>
int main( )
{
 FILE *fp;
char s[80];
int t;
clrscr( );
if(( fp=fopen("test.txt","w"))==NULL)
{
 printf("file cannot open\n");
 exit(1);
}
printf("enter a string and a numbers\n");
fscanf(stdin,"%s%d",s,&t);
fprintf(fp,"%s%d",s,t);
fclose(fp);
if((fp==fopen("test.txt","r"))=NULL)
{
 printf("cannot open file");
 exit(1);
}
fscanf(fp,"%s%d",s,&t);
fprintf(stdout,"%s%d",s,t);
return 0;
getch();
}
```

fread() and fwrite():

fread( ):-

*fread (void *pinarea, int elements size, int count,file-ptr);*

Here 'pinarea' refers to the input area in memory. Here generic pointer is used. This allows any pointer type to be passed to the functiom.

File read expects a pointer to the input area, which is usually a structure. This is because binary files are most often used to store structure.

The 'elements size' and 'count' are multiplied to determine how much data are to be transferred. The size is normally specified using the sizeof( )operator. Count is no of records in the structure.

The last parameters is the file pointer.

fwrite( ):-

Syntax:

*fwrite (void *poutarea, int elementsize, int count,file-ptr);*

fwrite() copies 'elementsize*count' bytes from the address specified by 'poutarea' to the file.

Here 'poutarea' refers to the output area in memory. Here generic pointer is used. This allows any pointer type to be passed to the function.

The size is normally specified using the sizeof( )operator. Count is no of records in the structure.

The last parameter is the file pointer.

Files

//Write a program that shows the example of fread and fwrite functions and read and write an array of structures.

```
#include<stdio.h>
#include<conio.h>
int main (void)
{
 FILE *fp;
 struct product
 {
    int cat_num;
    float cost;
  };
struct product p[3]={ { 3,30.2},{4,50.76},{5,70.9}};
struct product *k=&p;
fp=fopen("sample txt","w");
fwrite(p, sizeof(product),3,fp);
rewind(fp);
for(i=0;i<3,i++)
{
 fread (k,sizeof(product),1,fp);
printf("product %d;cat_num=%d,cost=%f\n",I,k→ ca_nam,k→cost);
}
getch( );
return 0;
}
```

File Status Functions:-

Test end of file (feof( )):-

Syntax: *feof(file-ptr);*

feof function is used to check if the end of the file has been reached. If used is at the end that is, if all the data have read the function returns nonzero (true). If the end of the file has not been reached zero (false) is returned.

Test error (ferror( )):-

Syntax: *ferror(file-ptr);*

Files

This function is used to check the error status file. Errors can be created for memory reasons. Ranging bad physical media to illogical operations, such as to read a file in the write state.

The ferror( ) returns true(nonzero) if an error has occurred. It returns false (zero) if no error has occurred.

```
//Write a program that illustrates error handling In file operations.
#include<stdio.h>
main( )
{
 char *filename;
 FILE *fp1, *fp2;
 int I,n;
fp1=fopen("sample.txt","w");
for( i=0; i<=100; i=i+100)
putw(I,fp1);
fclose(fp1);
printf("enter file name");
open-file;
scanf("%s", filename);
if((fp2=fopen (filename,"r"))= = NULL)
{
  printf("cannot open the file\n");
  printf("type file name again\n");
  goto open file;
 }
else
for(i=1;i<=20;i++)
{
 n=getw(fp2);
if(feof(fp2))
{
 printf("end of file\n");
 break;
}
else
printf("%d",n);
}
fclose(fp2);
}
```

Positioning Functions:-

Rewind file(rewind):-

Syntax:- *rewind (file-ptr);*

This function simply sets the file pointer to the beginning of the file.

Current position (ftell):-

Syntax:- *ftell(file-ptr);*

This function returns the current position of the file pointer in the file relative to the beginning of the file.

Example:

```
#include<stdio.h>
main( )
{
         FILE *fptr;
        long size;
        clrscr( );
        fptr = fopen("ABC.txt","r");
        if  (fptr = =NULL)
        perror ("enter opening file");
        else
        {
        fseek(fptr,0,SEEK_END);
        size=ftell(fptr);
        fclose(fptr);
        fclose(fptr);
        printf("size of ABC.txt=%ld bytes", size);
        }
}
```

Random Accessing a File:-

fseek( ):- the fseek( ) positions the file pointer to a specific position in a file.

Syntax: - *fseek (file ptr, long offset, int whereform);*

The first parameter is a position to opened file. The second parameter is a signed integer that specifies the number of bytes the pointer must more absolutely. The third parameter 'whereform' is three types.

SEEK_SET	0	Beginning of file
SEEK_CUR	1	Current position of file pointer
SEEK_END	2	End of file

This function returns zero value on success and nonzero value on failure.

Ex:-

```
#include<stdio.h>
main()
{
 FILE *fp;
fp=fopen("example.txt","w");
fputs("this is an apple",fp);
fseek(fp,9,SEEK_SET)
fputs("sam",fp)
fclose(fp);
return 0;
getch( );
}
```

Operations of fseek() Function:-

Statement	meaning
fseek(fp,0,0)	goto the beginning.
fseek(fp,0,1)	stay at the current position.
fseek(fp,0,2)	goto the end of file past the last character of the file.
fseek(fp,m,0)	more to(m+1) the bytes in the file.
fseek(fp,m,1)	go forward by m bytes.
fseek(fp,-m,1)	go backward by m bytes from the current position.
fseek (fp,-m, 2)	go backward by m bytes from the end.9postion the file to the file mth character from the end).

Files

/*Write a program to copy the contents of the one file to another file character by character*/

```
#include<stdio.h>
#include<conio.h>
main ( )
{
        FILE *fs,*ft;
        char ch;
        clrscr ( );
        fs=fopen (“prog1.c”,”r”);
        if (fs==NULL)
        {
                printf (“cannot open source file”);
                exit (1);
        }
        ft=fopen (“prog2.c”,”w”);
        if (ft==NULL)
        {
                printf (“cannot open target file”);
                fclose (fs);
                exit (1);
        }
        while (1)
        {
                ch=fgetc (fs);
                if (ch==EOF)
                        break;
                else
                        fputc (ch,ft);
        }
        fclose (fs);
        fclose (ft);
        return 0;
}
```

/* Write a program to count the number of character spaces, tabs and newline in files*/

```
#include<stdio.h>
#include<conio.h>
main ( )
{
 FILE *fp;
char ch;
int nol=0, not=0, noc=0;
fp=fopen (“source.dat”,”r”);
while (1)
{
 ch=fgetc (fp);
if(ch==EOF)
break;
noc++;
if (ch==' ')
nob++;
 if(ch=='\n)
 nol++;
 if(ch=='\t')
not++;
}
fclose(fp);
printf ("no of characters=%d\n", noc);
printf("no of blanks=%d\n", nob);
printf("no of tabs=%d\n", not);
printf("no of lines=%d\n", nol);
return 0;
}
```

Command Line Arguments:

All the programs we have been coded with no parameters for main but main is a function and as a function, it may have parameters when main has parameters they are known as command line arguments. These arguments allow the user to specify additional information when the program is invoked.

Command line arguments are two types. One an integer and the other an array of pointers to char (strings) that represents user-determined values to be passed to main.

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

argc - argument count

argv - argument vector

The first arguments the number of elements in the array identify in the second arguments. The value for the arguments is not entered using the keyboard, the system determines it from the arguments the user types.

The value of argc is determined from the user-typed values for argv.

// Program that demonstrate the use of command line arguments.

```
int main (int argc, char *argv)
{
        int i;
        printf("the no of arguments: %d\n", argc);
```

```
                printf(“the name of the program: %s\n”, argv [0]);
                for (i=1; i<argc; i++)
                printf (“user values no: %d, %s\n”, I, argv[i]);
                return 0;
        }
/*Write a program that copies one file into another by using command line
arguments..*/
#include<stdio.h>
#include<conio.h>
int main (int argc, char *argv [])
{
        FILE *fs,*ft;
        char ch;
        clrscr ();
        if(argc! =3)
        {
                puts (“improve number of arguments”);
                exit (1);
        }
        fs=fopen (argv[1],”r”);
        if (fs==NULL)
        {
                printf (“cannot open source file”);
                exit (1);
        }
        ft=fopen (argv[2],”w”);
        if (ft==NULL)
        {
                printf (“cannot open target file”);
                fclose (fs);
                exit (1);
        }

        while (1)
       {
        ch=fgetc (fs);
        if(ch==EOF)
                break;
        else
                fputc (ch,ft);
      }
        fclose (fs);
        fclose (ft);
return 0;
}
```