FILES

Definition : It is collection of records (or) It is a place on hard disk where data is stored permanently.

Types of Files: (1)Text file

(2)Binary File

1. Text File : It contains alphabets and numbers which are easily understood by human beings.

2. Binary file : It contains 1's and 0's which are easily understood by computers.

- Based on the data that is accessed, files are classified in to

(1) Sequential files

(2) Random access files

(1) Sequential files: Data is stored and retained in a sequential manner.

(2) Random access Files : Data is stored and retrieved in a random way.

Operations on files : 1. Naming the file

2. Opening the file

3. Reading from the file

4. Writing into the file

5. Closing the file

Syntax for opening and naming file.

1) FILE *File pointer;

Eg : FILE * fp;

2) File pointer = fopen ("File name", "mode");

Eg : fp = fopen ("sample.txt", "w");

```
FILE *fp;
fp = fopen ("sample.txt", "w");
```

Modes of the opening the file :

r	-	File is opened for reading
w	-	File is opened for writing
a	-	File is opened for appending (adding)
r+	-	File is opened for both reading & writing
w+	-	File is opened for both writing & reading
a+	-	File is opened for appending & reading
rt	-	text file is opened for reading
wt	-	text file is opened for writing
at	-	text file is opened for appending

r+t - text file is opened for reading & writing

w+t - text file is opened for both writing & reading

a+t - text file is opened for both appending & reading

rb - binary file is opened for reading

wb - binary file is opened for writing

ab - binary file is opened for appending

r+b - binary file is opened for both reading & writing

w+b - binary file is opened for both writing & reading

a+b - binary file is opened for both appending & reading.

1) Write mode of opening the file

```
FILE *fp;
fp =fopen ("sample.txt", "w");
```

a) If the file does not exist then a new file will be created

b) If the file exists then old content gets erased & current content will be stored.

2. Read mode of opening the file:

```
FILE *fp
fp =fopen ("sample.txt", "r");
```

a) If the file does not exists, then fopen function returns NULL value.

b) If the file exists then data is read from the file successfullly

3. Append mode of opening a file

```
FILE *fp;
fp =fopen ("sample.txt", "a");
```

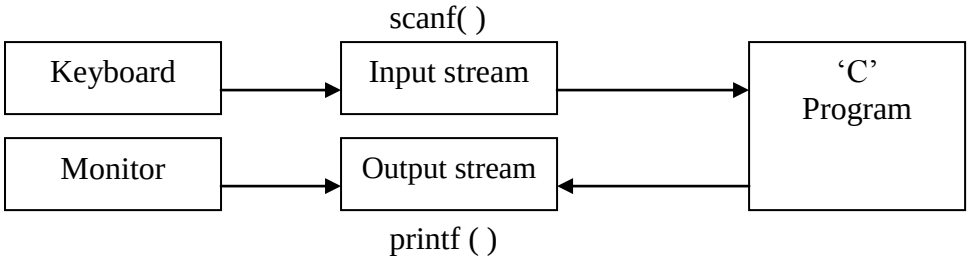
a) If the file doesn't exists, then a new file will be created.

b) If the file exists, the current content will be appended to the old content

Mode	Exist	Not exist
r	Read	fp = "NULL"
w	Current Content	New file will be created
a	Old content Current content	New file will be created

I/O STREAMS:

Stream : flow of data



I/0 functions:

1) high level I/o

- These are easily understood by human beings
- Advantage: portability.

2) Low level I/o

- These are easily understood by computer
- Advantages. Execution time is less
- Disadvantage: Non protability

High level I/o Functions

1) fprintf ( ) - to write data into a file
2) fscanf ( ) - To read data from a file
3) putc ( )/ fputc() - to write a character into a file
4) getc ( ) /fgetc() - to read a character from a file
5) putw ( ) - To write a number into a file
6) getw ( ) - To read number from a file
7) fputs ( ) - To write a string into a file
8) fgets ( ) - To read a string from a file
9)fread() - To read an entire record from a file
10)fwrite() - To write an entire record into a file

fprint ( ) & fscanf ( ) functions

1) fprint ( )

Syntax : fprintf (file pointer, " control string", variable list)

Eg: FILE *fp;

fprintf (fp, "%d%c", a,b);

2) fscanf ( )

Syntax : fscanf(file pointer, "control string", & variable list);

Eg: FILE *fp;

fscanf (fp, "%d%c", &a,&b);

Program for storing the details of an employee in a file and print the same

```
main ( )
{
        FILE *fp;
        int eno;
        char ename [30];
        float sal;
        clrscr ( );
        fp =fopen ("emp.txt", "w");
        printf ("enter the details of eno, ename, sal");
        scanf ("%d%s%f", &eno, ename, &sal);
        fprintf (fp, "%d%s%f", eno, ename, sal);
        fclose (fp);
        fp = fopen ("emp.txt", "r");
        fscanf (fp, "%d%s%f", &eno, ename, &sal);
        printf ("employee no: = %d", eno);
        printf ("employee name = %s", ename);
        printf ("salary = %f", sal);
        fclose (fp);
        getch( );
}
```

Program for storing the details of 60 employers in a file and print the same

```
main ( )
{
        FILE *fp;
         int eno, i;
        char ename [80];
        float sal;
        clrscr ( );
```

```
        fp = fopen (“emp1. txt”, “w”);
        for (i=1; i<60; i++)
        {
        printf (“enter the eno, ename, sal of emp%d”, i);
        scanf (“%d%s%f”, &eno, ename, &sal);
        fprintf (fp, “%d %s %f”, eno, ename, sal);
        }
        fclose (fp);
        fp = fopen (“emp1.txt”, “r”);
        for (i=1; i<60; i++)
        {
        fscanf(fp, “%d %s %f”, &eno, ename, &sal);
        printf (“details of employee %d are \n”, i);
        printf (“eno = %d, ename = %s, sal = %f”, eno, ename, sal);
        }
        fclose (fp);
        getch ( );
}
```

putc( ) and getc( ) functions:

1) putc ( ): It is used for writing a character into a file

Syntax :

```
putc (char ch, FILE *fp);
```

Eg :

```
FILE *fp;
char ch;
putc(ch, fp);
```

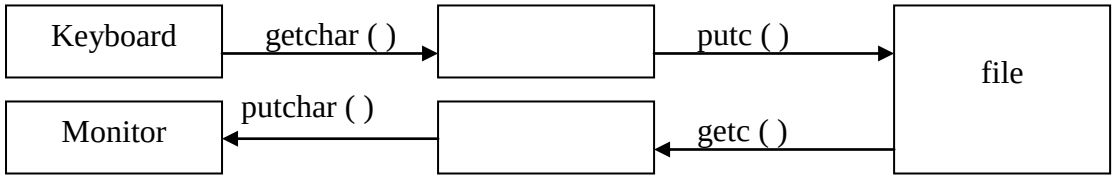
2) get c ( ) : It is used to read a character from file

Syntax :

```
char getc (FILE *fp);
```

Eg:

```
FILE *fp;
char ch;
ch = getc(fp);
```



Program :

```
main ( )
{
        FILE *fp;
        char ch;
        clrscr ( );
        fp = fopen ("characters.txt", "w");
        printf ("enter text. press ctrl+z at the end");
        while ((ch = getchar ( ))! = EOF)
        {
                putc(ch, fp);
        }
        fclose (fp);
        fp =open ("characters. txt", "r");
        printf ("file content is \n");
        while ((ch = getc (fp))! = EOF)
        {
                putchar (ch);
        }
        fclose (fp);
        getch ();
}
```

Output:

Enter text press ctrl+z at the end.
Hello how r u ^z
File Content is
Hello How r u

putw ( ) and getw ( ) functions:

1. putw( ) : It is used for writing a number into file.

Syntax: putw (int num, FILE *fp);

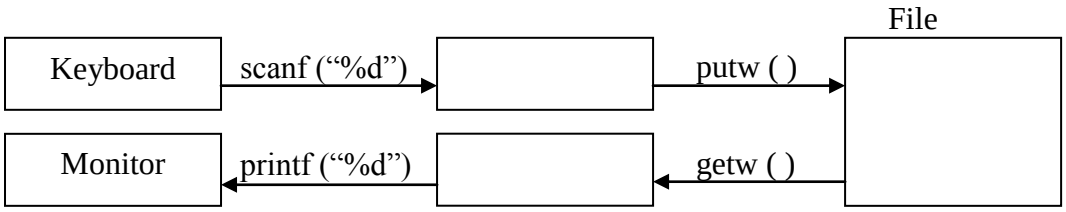
Eg: FILE *fp;
int num;
putw(num, fp);

2. getw ( ): It is used for reading a number from a file

Syntax :

int getw (FILE *fp);

Eg : FILE *fp;
int num;
num = getw(fp);



Program for storing no’s from 1 to 10 and print the same

```
main ( )
{
        FILE *fp;
        int i;
        clrscr ( );
        fp = fopen (“number. txt”, “w”);
        for (i =1; i< = 10; i++)
        {
                putw (i, fp);
        }
        fclose (fp);
        fp =fopen (“number. txt”, “r”);
        printf (“file content is ”);
        for (i =1; i< = 10; i++)
        {
                i= getw(fp);
                printf (“%d”,i);
```

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

Program for copying the contents of one file into another file

```
main ( )
{
    FILE *fp1, *fp2;
    char ch;
    clrscr ( );
    fp1 = fopen ("file1.txt", "w");
    printf ("enter text press ctrl+z at the end");
    while ((ch = getchar ( ))! = EOF)
    {
        putc(ch, fp1);
    }
    fclose (fp1);
    fp1 =fopen ("file1. txt", "r");
    fp2 =fopen ("file2. txt", "w");
    while ((ch = getc (fp1))! = EOF)
    {
        putc(ch,fp2);
    }
    fclose (fp1);
    fclose (fp2);
    fp2 = fopen ("file2.txt", "r");
    printf ("File2 contents are");
    while ((ch = getc(fp2))! = EOF)
        putchar (ch);
    fclose (fp2);
    getch ();
}
```

Program for displaying the contents of a file

```
main ( )
{
        FILE *fp;
        char ch ;
        clrscr ( );
        fp = fopen (“file1.txt”,”r”);
        if (fp = = NULL)
        {
                printf (“File does not exist”);
        }
        else
        {
                printf (“file content is”)
                while ((ch = getc(fp))! = EOF)
                putchar (ch);
        }
        fclose (fp);
        getch ( );
}
```

Program to merge two files into a third file. (the contents of file1, file2 are placed in file3)

```
main ( )
{
        FILE *fp1, *fp2, *fp3;
        char ch;
        clrscr ( );
        fp1 = fopen (“file1.txt”, “w”);
        printf (“enter text into file1”);
        while ((ch = getchar ( ))! = EOF)
        {
                putc(ch, fp1);
        }
        fclose (fp1);
        fp2 = fopen (“file2.txt”, “r”);
```

```
        printf ("enter text into file2");
        while ((ch = getchar ( ))! = EOF)
                putc(ch, fp2);
        fclose (fp2);
        fp1 =fopen ("file1. txt", "r");
        fp2 =fopen ("file2. txt", "r");
        fp3 =fopen ("file3. txt", "w");
        while ((ch = getc (fp1))! = EOF)
                putc(ch,fp3);
         while ((ch = getc (fp2))! = EOF)
                putc(ch,fp3);
        fclose(fp1);
        fclose (fp2);
        fclose (fp3);
        fp3 = fopen ("file3.tx", "r");
        printf ("File3 contents is");
        while ((ch = getc(fp3))! = EOF)
                purchar (ch);
        fclose (fp3);
        getch ();
}
```

fput c ( ) and fgetc ( ) functions :

1) fputc( ) : It is used for writing a character in to a file .

Syntax :

```
fputc (char ch, FILE *fp);
```

Eg :

```
        FILE *fp;
        char ch;
        fputc (ch.fp);
```

2. fgetc( ) : This is used for reading a character from a file

Syntax :

```
fputc (char ch, FILE *fp);
```

Eg :

```
        FILE *fp;
        char ch;
```

```
ch = fgetc(fp);
```

fgets ( ) and fputs ( ) functions :

1) fgets ( ) : It is used for reading a string from a file

Syntax :

fgets (string variable, No. of characters, File pointer);

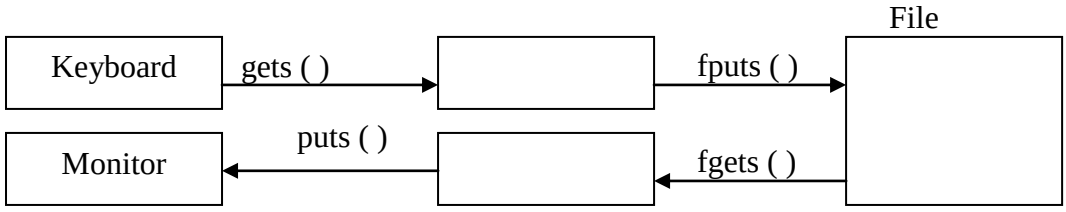
```
Eg :    FILE *fp;
        char str [30];
        fgets (str,30,fp);
```

2) fputs ( ) : It is used for writing a string into a file

Syntax :

fputs (string variable, file pointer);

```
Eg :    FILE *fp;
        char str[30];
        fputs (str,fp);
```



Program :

```
main ( )
{
        FILE *fp;
        char str [30];
        int i,n;
        clrscr ( );
        printf ("enter no of strings");
        scanf ("%d", & n);
        fp = fopen ('strings.txt", "w");
        for (i=1; i<=n; i++)
        {
                printf ("enter string %d",i);
                gets (str);
                fputs (str, fp);
```

```
        }
        fclose (fp);
        fp = fopen (”strings.txt”, ”r”);
        for (i=1; i<=n; i++)
        {
                fgets (str, 30, fp);
                printf (”string %d =”, i);
                puts (str);
        }
        fclose (fp);
        getch ( );
}
```

fread ( ) and fwrite ( ) functions

1. fread ( ) : It is used for reading entire record at a time.

Syntax : fread(& structure variable, size of (structure variable), no of records, file pointer);

Eg :
```
struct emp
        {
                int eno;
                char ename [30];
                float sal;
        } e;
        FILE *fp;
        fread (&e, sizeof (e), 1, fp);
```

2. fwrite ( ) : It is used for writing an entire record at a time.

Syntax : fwrite(& structure variable , size of structure variable, no of records, file pointer);

Eg :
```
struct emp
        {
                int eno:
                char ename [30];
                float sal;
        } e;
        FILE *fp;
        fwrite (&e, sizeof(e), 1, fp);
```

program for storing the details of 60 students into a file and print the same using fread ( ) and fwrite ( )

```
struct student
{
        int sno;
        char sname [30];
        float marks;
};
main ( )
{
        struct student s[60];
        int i;
        FILE *fp;
        clrscr ( );
        fp = fopen (“student1. txt”, “w”);
        for (i=0; i<60; i++)
        {
                printf (“enter details of student %d”, i+1);
                scanf (“%d%s%f”. &s[i].sno,s[i].sname, &s[i].marks);
                fwrite (&s[i], sizeof (s[i]), 1, fp);
        }
        fclose (fp);
        fp = fopen (“student1. txt”, “r”);
        for (i=0; i<60; i++)
        {
                printf (“details of student %d are”, i+1);
                fread (&s[i], sizeof (s[i]) ,1,fp);
                printf(“student number = %d”, s[i]. sno.);
                printf(“student name  = %s”, s[i]. sname.);
                printf(“marks = %f”, s[i]. marks);
        }
        fclose (fp)
        getch( );
}
```

ERROR HANDLING IN FILES:-

- Some of the errors in files are
  1. Trying to read beyond end of file
  2. Device over flow
  3. Trying to open an invalid file
  4. Performing a invalid operation by opening a file in a different mode.

Functions for error handling.

1) ferror ( )
2) perror ( )
3) feof ( )

1. ferror ( )

It is used for detecting an error while performing read / write operations.

Syntax :

int ferror (file pointer);

eg : FILE *fp;
if (ferror (fp))
printf ("error has occurred");

→ it returns zero if success and a non- zero otherwise.

2. perror ( )

→ It is used for printing an error.

Syntax :

perror (string variable);

Eg : FILE *fp;
char str[30] = "Error is";
perror (str);

O/P : Error is : error 0

Program :

```
main ( )
{
        FILE *fp;
        char str[30] = "error is";
        int i = 20;
        clrscr ( );
        fp = fopen ("sample. txt", "r");
        if (fp = = NULL)
        {
                printf ("file doesnot exist");
        }
        else
        {
        fprintf (fp, "%d", i);
        if (ferror (fp))
        {
                perror (str);
                printf ("error since file is opened for reading only");
        }
fclose (fp);
getch ( );
}
```

O/P: Error is : Error1 → compiler generated.

Error since file is opened for reading → by us.

3. feof ( )

It is used for checking whether end of the file has been reached (or) not.

Syntax :

```
int feof ( file pointer);
```

Eg :

```
FILE *fp;
if (feof (fp))
printf ("reached end of the file");
```

⟶ If returns a non zero if success and zero otherwise.

Program:

```
main ( )
{
        FILE *fp;
        int i,n;
        clrscr ( );
        fp = fopen (“number. txt”, “w”);
        for (i=0; i<=100;i= i+10)
        {
                putw (i, fp);
        }
        fclose (fp);
        fp = fopen (“number. txt”, “r”);
        printf (“file content is”);
        for (i=0; i<=100; i++)
        {
                n = getw (fp);
                if (feof (fp))
                {
                printf (“reached end of file”);
                break;
                }
                else
                {
                printf (“%d”, n);
                }
        }
        fclose (fp);
getch ( );
}
```

Outpute : File content is

10 20 30 40 50

60 70 80 90 100

Reached end of the file.

Other file functions

Random accessing of files

1. ftell ( )
2. rewind ( )
3. fseek ( )

1. ftell ( ) : It returns the current postion of the file pointer

Syntax : int n = ftell (file pointer)

Eg : FILE *fp;

int n;

_____

_____

_____

n = ftell (fp);

Note : ftell ( ) is used for counting the no of characters entered into a file.

2. rewind ( )

It makes the file pointer move to the beginning of the file.

Syntax: rewind (file pointer);

Eg : FILE *fp;

---

rewind (fp);
n = ftell (fp);
printf ("%d", n);

o/p: 0 (always).

3. fseek ( )

It is used to make the file pointer point to a particular location in a file.

Syntax: fseek(file pointer,offset,position);

offset :

- The no of positions to be moved while reading or writing.
- If can be either negative (or) positive.

→ Positive - forward direction.

→ Negative – backward direction .

position :

- it can have 3 values.

0 – Beginning of the file

1 – Current position

2 – End of the file

Eg :

1. fseek (fp,0,2) - fp is moved 0 bytes forward from the end of the file.
2. fseek (fp, 0, 0) – fp is moved 0 bytes forward from beginning of the file
3. fseek (fp, m, 0) – fp is moved m bytes forward from the beginning of the file.
4. fseek (fp, -m, 2) – fp is moved m bytes backward from the end of the file.

Errors :

1. fseek (fp, -m, 0);
2. fseek(fp, +m, 2);

Write a program for printing some content in to the file and print the following ?

1. Number of characters entered into the file.
2. Reverse the characters entered into the file.

```
main ( )
    {
    FILE *fp;
    char ch;
    int n;
    clrscr ( );
    fp = fopen (“reverse. txt”, “w”);
    printf (“enter text press ctrl+z of the end”);
    while ((ch = getchar( ) ) ! EOF)
            {
                    putc (ch, fp);
            }
    n = ftell (fp)
    printf ( “No. of characters entered = %d”, n);
    rewind (fp);
    n = ftell (fp);
    printf (“fp value after rewind = %d”,n);
    fclose (fp);
    fp = fopen (“reverse.txt”, “r”);
    fseek (fp, -1, 2);
```

```
printf ("reversed content is");
do
{
        ch = getc (fp);
        printf ("%c", ch);
} while (!fseek (fp, -2, 1);
fclose (fp);
getch ( );
}
```

Output : Enter text press ctrl z at the end.

```
        How are you ^z
No. of characters entered = 11
fp value after rewind =0
```

Reversed content is **uoy era woh**.

Command line arguments

- Arguments given at command prompt.
- main ( ) takes 2 arguments.

1) int argc – argument count.

2) char *argv [ ] – argument vector.

Program :

```
main (int arg c, char * argv [])
{
        int i;
        clrscr ( );
        printf ("no. of arguments = %d", argc);
        printf ("arguments given at cmd prompt are");
        for (i=0; i<argc; i++)
        {
                printf ("%s \t", argv [i]);
        }
getch ( );
}
```

Program : for reversing characters in a file given at command prompt

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

```
{
        FILE *fp;
        char ch;
        clrscr ( );
        fp = fopen (argv[1], "w");
        printf ("enter text press ctrl+z at the end");
        while ((ch = getchar ( ))! = EOF)
                putc (ch, fp);
        fclose (fp);
        fp  = fopen (argv[1], "r");
        fseek ( fp, argv[2], 0);
        do
        {
                ch = getc (fp);
                putchar (ch);
        } while (! fseek (fp, -2, 1));
        fclose (fp);
        getch ( );
}
```