CHAPTER 12

FILE MANAGEMENT IN C

In real-life problems involve large volumes of data and in such situations, the console oriented I/O operations pose two major problems,

1) It becomes difficult and time consuming to handle large volumes of data through terminals.
2) The entire data is lost when eithere the prog terminates or the computer is turned off.

Therefore it becomes necessary to store data on disks and read whenever required, without destroying the data. This employs the concept of files to store data.
A file is a place on disk where a group of related data is stored. Basic file operations are,

1) naming a file
2) opening a file
3) reading data from a file
4) writing data to a file
5) closing a file

There are two way to perform file operations

1) Low level I/O and uses unix system calls.
2) High level I/O and uses functions in C's stdndard I/O Library.

High Level I/O functions are,

fopen() => creates a new file for use
fclose() => closes a file which is opened
fprintf() => prints to a file
fscanf() => scans from a file
fseek() => set a position to a desired point in file
ftell() => gives current position in file

Defining and Opening a File:

If we want to store data in file in secondary memory, we must specify certain things about the file to the OS, they include,

*) file name
*) data structure => FILE is a datatype
*) purpose => mode of operation

syntax:

FILE *fptr;

fprt = fopen (“filename”,”mode”);

where ,

fptr -> file pointer

mode ->

r-> open in readonly

w-> open in write only

a-> open in append mode

ex:

FILE **fp1,**fp2;

Fp1 = fopen(“data”,”r”);

Fp2=fopen(“result”,”w”);

Closing a file

Syntax :

Fclose(file pointer);

Ex:

Fclose(fp1);

Fclose(fp2);

I/O OPERATIONS ON FILES:

c = getc(fp2);

putc(c,fp1);

getc() returns EOF when end of file has reached.

*)prog to read data from keyboard and write to a file called INPUT again read the same data from the INPUT file and display on screen.

```
main()
{
        FILE *fp;
        Char c;
        printf("DATA I/P");
        fp = fopen("INPUT","w");
        while((c = getchar()) != EOF)
                putc(c,fp);
        fclose(fp);
        printf("\n DATA O/P");
        fp = fopen("INPUT","r");
                while((c = getchar()) != EOF)
                        printf("%c",c);
        fclose(fp);
}/* EOM */
```

Random Access To files

_Ftell (file ptr): take a file ptr and returns a no of type long, that corresponds to current position. This f ^ usefull in saving the current position of a file, each can be used later in a prog.

Ex:- n=ftell(fp);

Fseek ( ) :-when we are interested to access only a specific part of a program, it is necessary to lcate that desired part in a file before data can be read or written . to do so file ptr should move to that PT this is done by using fseek.

Syn tan :-fseek(file ptr ,offset,position)

Offset→ number of bytes from a specified position in a file .it is of type long.

Position→it indicates position from which it should start moving. It can take one of three values, 0,1or 2.

0→SEEK_SET =>move the ptr from bof

1→SEEK_CUR=>move the ptr with ref to its current position.

2→SEEK_END=>move ptr from EOF.

Ex:-fseek(fp,10L,0)
=>moves ptr 10bytes from BOF

```
*)prog
#define MAX 15
main ( )
{     FILE *fptr;
int I=0, letter ;
char *file name ;
long position ;
char  letter;
printf(“enter the file name \ n”);
gets(file name);
fptr = fopen(file name, “r”);
if (fptr = = NULL){printf(“cant open file /n”)
                        exit(0);}
position= ftell(fptr);
printf(“\n file ptr is positioned at %\d th bytes\n”,position)}
while (I<MAX){
if(letter = fgetc (fptr))!=EOF)
{printf (“%C”, letter);}
I=I+1;
}
position = ftell (fptr);
fseek(fptr,-- 10L, 1);
positon = ftell (fptr);
}
/*EOM */
```