FILE HANDLING

FILE HANDLING

- PROBLEM:-if we need same data to be processed again and again,we have to enter it again and again and if volume of data is large,it will be time consuming process.
- SOLUTION:-data can be permanently stored,a program can read it directly at a high speed.
- The permanent storage space on a disk is called FILE.

- FILE is a set of records that can be accessed through the set of library functions.

FILE TYPES

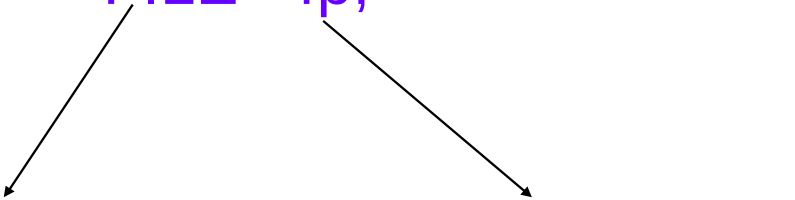
1.SEQUENTIAL FILE:-in this data is kept permanently.if we want to read the last record of file then we need to read all the records before that record.it means if we wan to read 10^{th} record then the first 9 records should be read sequentially for reaching to the 10^{th} record.

2.RANDOM ACCESS FILE:- in this data can be read and modified randomly.if we want to read the last record of the file,we can read it directly.it takes less time as compared as sequential file.

HOW TO DEFINE A FILE

Before using a file in a program,we must open it,which establishes a link between the program and the operating system.A file must be defined as:-

```
FILE  *fp;
```



where FILE is the data structure which is included in the header file,STDIO.H and fp is declared as a pointer which contains the address of a character which is to be read.

Operation on a file

fp = fopen (“ABC.DOC”, “r”);

Here fp =file pointer

fopen= function to open a file

ABC.DOC=name of the file

r=mode in which file is to be opened

fopen is a function which contains two arguments:-

1. File name ABC.DOC
2. Mode in which we want to open the file.
3. Both these are of string types.

fopen():-

- This function loads the file from disk to memory and stores the address of the first character to be read in the pointer variable fp.
- If file is absent,it returns a NULL pointer variable.

fclose(fp):- the file which is opened need to be closed and only can be done by library function fclose.

File opening modes

1. Read mode(r):- purpose to read from the file.if file exists,it sets up a pointer which points to first char in it and if it does not exist,it returns NULL.
2. Write mode(w):-purpose to write in the file.if file exists,its contents are overwritten.if it does not exist new file is created.
3. Append mode(a):-purpose is to add data in already existing file.if it does not exist new file is created.

4)Read +(r +):-it is an extension to read mode.the operations possible on this mode are to read existing,writing new contents and modifying existings contents.it returns NULL if file does not exist.

5)Write +(w +):-it is similar to write mode, mode.the operations possible on this mode are to write new contents and modify contents already written.

6)Append +(a +):- it is similar to append mode,the operations possible on this mode are to read existing contents,add new contents at the end of file but cannot modify existing contents.

```
main()
{
FILE *fp;
fp=fopen(“data.txt”,”r”);
If(fp = = NULL)
{
printf(“cannot open file”);
}
exit(0);
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
   FILE *fptr;
   clrscr();
   fptr=fopen("a.txt",“w");
   if(fptr==null)
        printf("\n file cannot be opened for creation");
   else
        printf("\n file created successfully");
   fclose(fptr);
}
```

- Binary modes:-

- wb(write):-it opens a binary file in write mode.

```
fp=fopen(“data.dat”,”wb”);
```

Here data.dat file is opened in binary mode for writing.

- rb(read):-it opens a binary file in read mode.

```
fp=fopen(“data.dat”,”rb”);
```

Here data.dat file is opened in binary mode for reading.

```
main()
{
FILE *fp;
char c=‘ ‘;
fp=fopen(“data.dat”,”wb”);
if(fp= =NULL)
{
printf(“can not open file”);
exit(0);
}
printf(“write data & to stop press ‘.’ ”);
while(c!=‘.’)
{
c=getche();
fputc(c,fp);
}
fclose(fp);
fp=fopen(“data.dat”,”rb”);
printf(“contents read”);
while (! Feof(fp))
printf(“%c”,getc(fp));
}
```

Reading from a file

- Different functions to read a char are fgetc and getc.both perform similar functions and have the same syntax.

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

char type variable — function name — file pointer

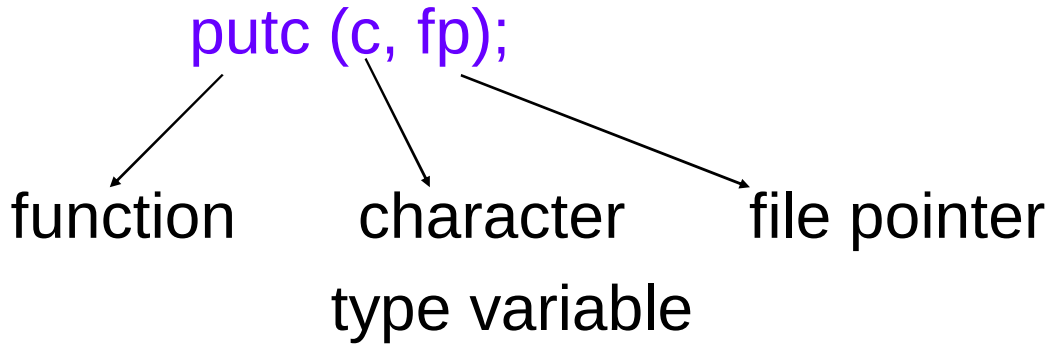
- fgets() or gets():-used to read strings from a file and copy string to a memory location which is referenced by array.

```
fgets(str, n, fptr);
```

array of chars — max.length — file pointer

Writing chars to a file

- fputc() or putc():-both these fuction write the character in the file at the position specified by pointer.



The file pointer automatically moves 1 position forward after printing characters in the file.

Read &write char into file and prints them

```
#include<conio.h>
void main()
{
    FILE *fptr;
    char c;
    clrscr();
    fptr=fopen("a.txt","r");
    printf("the line of text is");
    while((c=getc(fptr))!EOF)
     {
         printf("%c",c);
      }
    fclose(fptr);
    getch();
}
```

```
#include<conio.h>
#include<stdio.h>
void main()
{
    FILE *fptr;
    char c;
    clrscr();
    fptr=fopen("a.txt","w");
    printf("Enter the line of text,press ^z
    to stop ");
      while(c=getchar()!EOF)
      {
           putc(c,fptr);
      }
    fclose(fptr);
}
```

Program to understand use of fgetc()

```
main( )
{
FILE *fptr;
char ch;
fptr= fopen(“rec.dat”,”r”);
if(fptr = = NULL)
printf(“file doesn’t exist”);
else
{
while( ch = fgetc(fptr) !=EOF)
printf(“%c”,ch);
}
fclose(fptr);
}
```

EOF:-END OF FILE

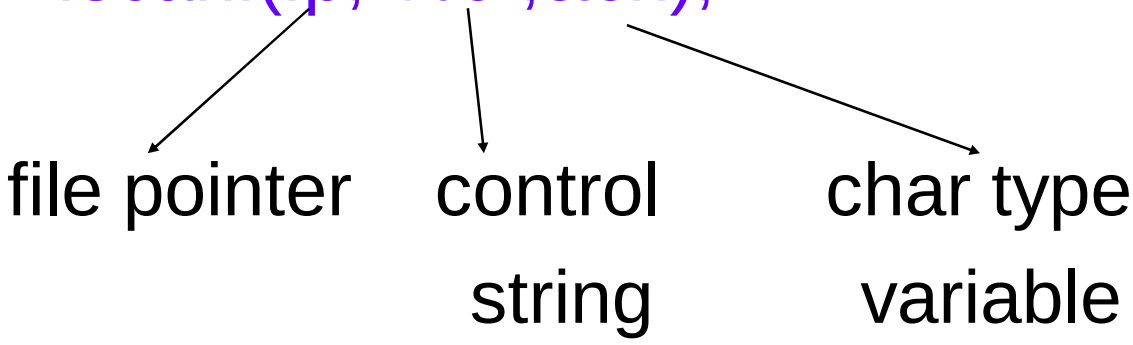
EOF is an integer value sent to prog.by OS.its value is predefined to be -1 in STDIO.H, No char with the same value is stored on disk.while creatig file,OS detects that last character to file has been sent,it transmits EOF signal.

Writing to file using fputs(sptr,fptr)

```
main()
{
FILE *fp;
char name[20],arr[50];
printf(“enter the file name”);
scanf(“%s”,name);
fp=fopen(name,”w”);
if(fp = = NULL)
{
printf(“file can’t be opened”);
exit(0);
}
else
{
printf(“the string is “);
gets(arr);
fputs(arr,fp);
}
fclose(fp);
}
```

- fscanf( ):- it is used to read to any type of variable i.e.,char,int,float,string

fscanf(fp,”%c”,&ch);



fscanf(fp,”%c%d%f”,&a,&b,&c);

We write stdin,this func call will work similar simple scanf

Because stdin means standard i/p i.e.keyboard.

fscanf(stdin,”%c”,&ch);

Is equivalent to

scanf(“%c”,&ch);

```
#include<conio.h>
void main()
{
FILE  *fptr;
char name[10];
int sal;
fptr=fopen(“rec.dat”,”r”);
fscanf(fptr,”%s%d”,name,&sal);
while(!feof(fptr))
{
printf(“%s%d”,name,sal);
fscanf(fptr,”%s%d”,name,&sal);
}
    fclose(fptr);
    getch();
}
```

- fprintf():- it is used to print chars,numbers or strings in a file.

fprintf (fp,“%d”,a);

With this function,we can print a no. of variables with single call.

fprintf (fp,”%c %d %d”,a,b,c);

We write stdout,this func call will work similar simple printf

Because stdout means standard o/p i.e.keyboard.

fprintf(stdout,”%c”,ch);

Is equivalent to

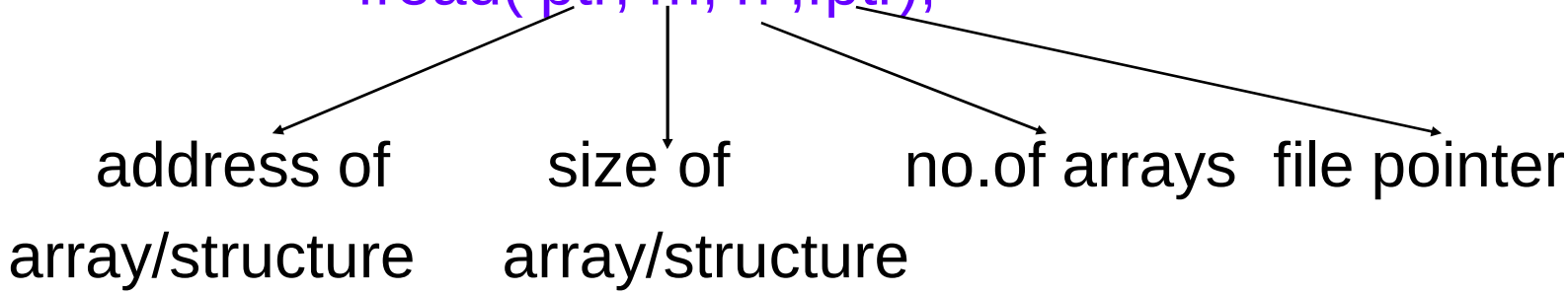
printf(“%c”,ch);

Program for fprintf()

```
main()
{
FILE  *fp;
char name[10];
fp=fopen(“rec.dat”,”w”);
printf(“enter your name”);
scanf(“%s”,name);
fprintf(fptr,”%s”,name);
fclose(fp);
}
```

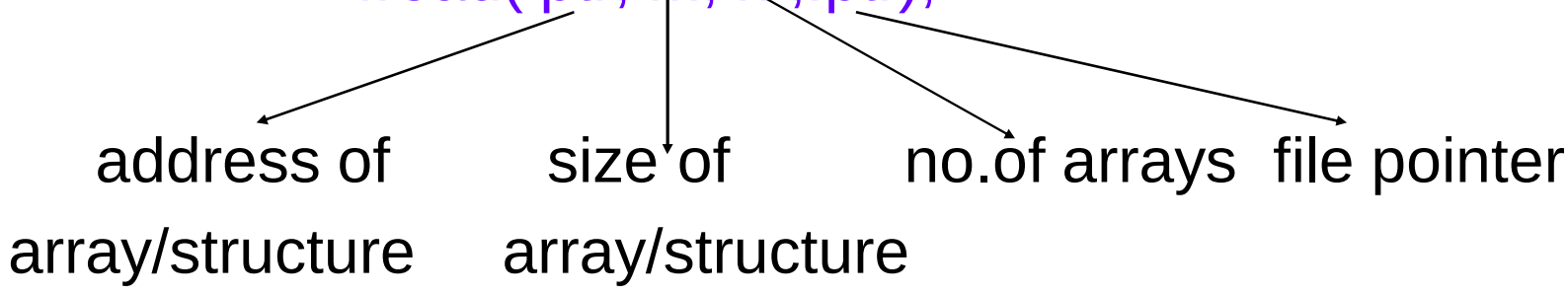
- fread():-reads a block(array or structure)from a file.

fread(ptr, m, n ,fptr);



- fwrite():-writes a block(array or structure)from a file.

fread(ptr, m, n ,fptr);



Prog for fread()

```
#include<conio.h>
struct student
{
    int rollno;
    char name[20];
};
void main()
{
    struct student st;
    file *fptr;
    fptr=fopen("d.txt","r");
    clrscr();
    fread(&st,sizeof(st),1,fptr);
    printf("roll number is %d",st.rollno);
    printf("name is %s",st.name);
    fclose(fptr);
    getch();
}
```

Prog for fwrite()

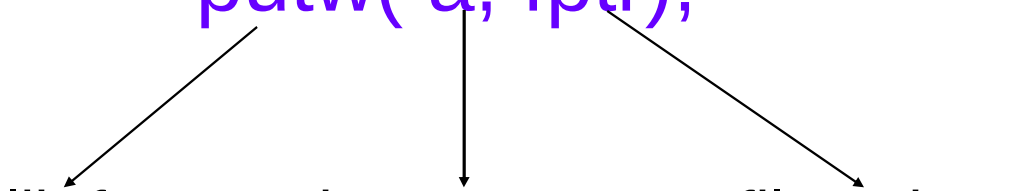
```
#include<conio.h>
struct student
{
    int rollno;
    char name[20];
};
void main()
{
    struct student st;
    file *fptr;
    fptr=fopen("d.txt","w");
    clrscr();
    printf("enter roll number");
    scanf("%d",&st.number);
    printf("enter name");
    gets(st.name);
    fwrite(&st,sizeof(st),1,fptr);
    fclose(fptr);
    getch();
}
```

- getw():-it is an integer oriented function.it is used to read an integer from file in which numbers are stored.

a=getw(fptr);

- putw():-it prints a number in the file.

putw(a, fptr);



Prog for getw()

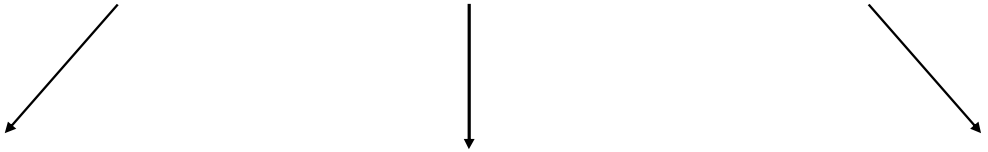
```
#include<conio.h>
void main()
{
    file *fptr;
    int i,n;
    clrscr();
    fptr=fopen("b.txt","r");
    printf("\n the number are")
      for(i=1;i<=10;i++)
      {
          n=getw(fptr);
          printf("%d\t",n);
    }
    fclose(fptr);
getch();
}
```

Prog for putw()

```
#include<conio.h>
void main()
{
    file *fptr;
    int i,n;
    clrscr();
    fptr=fopen("b.txt","w");
        for(i=1;i<=10;i++)
        {
    printf("enter number");
    scanf("%d",&n);
    putw(n,fptr);
    }
    fclose(fptr);
    getch();
}
```

- fseek():-it is used for setting the file position pointer at the specified byte.

```
fseek( FILE *fp,long displacement,int  origin);
```



file pointer

long int can be +ve/-ve

no.of bytes skipped backward(-ve) forward(+ve)

Constant	Value	position
SEEK_SET	0	Beginning of file
SEEK_CURRENT	1	Current position
SEEK_END	2	End of file

fseek(p,10L,0);

origin is 0,means that displacement will be relative to beginning of file,so position pointer is skipped 10 bytes forward from beginning of file.2nd argument is long integer so L is attached with it.

fseek(p,8L,SEEK_SET);

position pointer is skipped 8 bytes forward from beginning of file.

fseek(p,-6L,SEEK_END);

position pointer is skipped 6 bytes backward from the end of file.

Replace a char x by char y

```
main()
{
FILE *fp;
char temp,name;
printf(“enter name of file”);
scanf(“%s”,name);
fp=fopen(“name”,”r”);
while((temp=getc(fptr))!=EOF)
{
if(temp= =‘x’)
{
fseek(fp, -1,1);
putc(‘y’,fp);
}
}
fclose(fp);
}
```

ftell():-it is required to find the current location of the file pointer within the file

ftell(fptr);

Fptr is pointer for currently opened file.

```
main()
{
FILE *fp;
char ch;
fp=fopen(“text”,”r”);
fseek(fp,241,0);
ch=fgetc(fp);
while(!feof (fp))
{
printf(“%c”,ch);
printf(“%d”,ftell(fp));
ch=fgetc(fp);
}
fclose(fp);
}
```

- Rewind():-it is used to move the file pointer to the beginning of the given file.

rewind(fptr);

```
main()
{
FILE *fp;
fp=fopen(“stu.dat”,”r”);
if(fp==NULL)
{
printf(“file can not open”);
exit(0);
}
printf(“position pointer %d”,ftell(fp));
fseek(fp,0,2);
printf(“position pointer %d”,ftell(fp));
rewind(fp);
fclose(fp);
}
```

- ferror():-it is used for detecting error in the file on file pointer or not.it returns the value nonzero if an error occurs otherwise returns zero value.

```
ferror(fptr);
```