File Handling:

File:

A file is a collection of data stored on secondary device such as hard disk.

Advantages of using files:

→ It is time consuming to input large amount of data through keyboard so ~~input~~ instead of that input data can be stored in a file & program can read from this file.

Text file:

a text file is one, in which data is stored as sequence of characters. that can be processed sequentially.

~~Various steps involved when we do file manipulations:~~

Modes of a file:

mode of a file indicates whether file is opened for ~~read~~ reading or writting or appending.

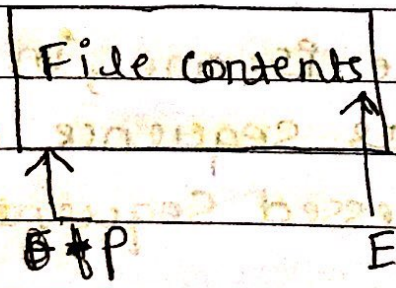
The various modes are

mode

i) r → open a text file for reading
ii) w → create a file for writting
iii) a → Append to a text file.

i) r (read mode):

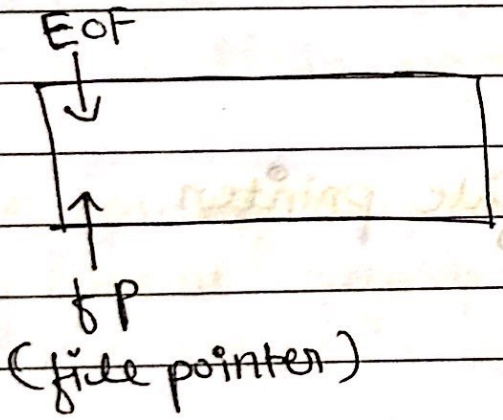
This mode is used for opening an existing file to perform read operation.



→ here fp is a file pointer pointing at beginning of the file
→ EOF indicates End of file.
→ and if file does not exists, an error is returned.

ii) w (write mode):

This mode is used to creat a file. If file does not exist with given name then file is created otherwise if file already existe original file contents are lost.



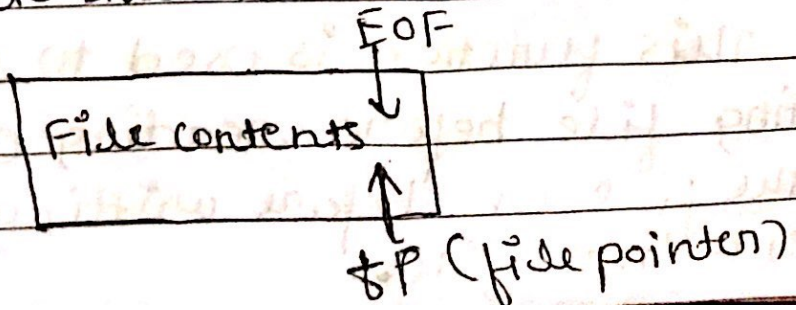
→ both EOF & file pointer (fp) points at the beginning of the file

iii) a (append file):

This mode is used to insert the data at the end of the existing file.

→ if file does not existe, a file is created.

→ If the file already exists, the file pointer points to the end.



Steps in using file or in file manipulation

1] Declare a file pointer
2] open a file
3] Read data from the file or write the data into file.
4] close the file.

1] Declaring a file pointer

```
FILE *fp;
```

→ here FILE is a derived data type which is already defined in <stdio.h>
→ fp is file pointer.
→ The fp (file pointer) variable can be used either as local or global variable.

2] File open & close functions:

* fopen():

This function is used to open existing file before reading contents of the file or before writting some contents into file.

Syntax for fopen() function

Syntax:

```
FILE *fp;

fp = fopen(filename, mode);
```

→ fp is a file pointer of type FILE
→ filename is the name of the file to be opened.
→ mode can be "r", or "w", or "a".
→ This fopen() function returns NULL if some error in opening a file.

Example:

```
FILE *fp;

fp = fopen("abc.txt", "r");
```

→ abc.txt is name of the file to be opened & r (read mode) indicates file need to opened in read mode.

* fclose():

This function is used to close the file after performing read or write operations.

Syntax:

```
fclose(fp);
```

→ fp is file pointer.

→ This function returns EOF if some error in closing file otherwise returns 0 if successfull.

Example:

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

INPUT & OUTPUT operations in files:

→ Input operations means reading contents of the file

→ output operation means writting some contents to the file.

Following functions are used for INPUT & OUTPUT (Reading / writting) operations

i) fscanf()
ii) fprintf()
iii) fgets()
iv) fputs()
v) fgetc()
vi) fputc()

i) fscanf():

The fscanf() function is used to get data from file, pointed by fp (file pointer). or to read data from file.

→ The Difference between scanf() function & fscanf() is that scanf() is used to read from keyboard where as fscanf() is used to read data from file pointed by fp.

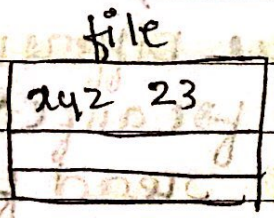
Syntax:

```
fscanf(fp, "format string", list);
```

→ fp is a file pointer, it can point to source file.
→ format string consist of format specifiers such as %d, %s, %f etc.
→ list is variable list, in which data obtained from file is stored.

Example:
if we want to read name & rollno of a student from file then following fscanf() function can be used.

```
int rollno; char name[20];
fscanf(fp, "%s %d", name, &rollno);
```



ii) fprintf():

fprintf() function is used to send data to output file, pointed by fp. or to write data to a file.

→ The Difference between printf() & fprintf() is that printf() is used to display data on screen (monitor) & fprintf() is used to send data to the file.

Syntax:

fprintf(fp, "format string", list);

→ fp is a file pointer, which is pointing to the file that has been opened for writting.

→ format string consist of format specifiers such as %d, %s, %f etc.

→ list is variable list, who's values need to be sent to file, pointed by fp.

Example: suppose we want to store name & rollno of a student into file then following fprintf() function is used

```
int rollno=1;
char name[20] = "xyz";

fprintf(fp, "%d %s", rollno, name);
fprintf(stdout, "%d %s", rollno, name);
```

→ fp is a file pointer that is pointing to the file which has been opened for writting.
→ then rollno & name values i,e 1 & xyz are stored in file pointed by fp.
→ if we use stdout in in above function then contents of rollno, name are displayed on screen.

iii) fgets():

fgets() function is used to read string or sequence of characters from file, pointed by fp.

Syntax:

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

→ fp is file pointer, which is pointing to file that is opened for reading
→ n indicates number of characters to be read from file.

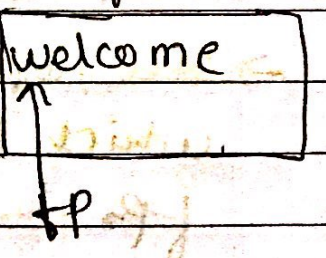
→ str is the string name, in which the characters read from file are stored.

→ This function Returns NULL if read operation is unsuccessful.

Example:

if we have string "welcome" stored in a file then following fgets() function reads first 3 characters from file & stores them in str string.

```
fgets(str, 3, fp);
```



iv) fputs():

fputs() function is used to store string or sequence of characters into a ~~specified~~ file pointed by fp.

Syntax:

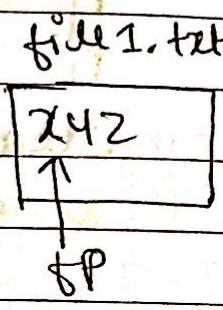
```
fputs(str, fp);
```

→ fp is a file pointer which is pointing to a file that is opened for writing.

→ str is a string name who's value is stored ~~on file~~ in file pointed by fp.

Example:

```
char name[7] = "xyz";
fputs(name, fp);
```



→ The content of string "name" is xyz which is stored in a file pointed by fp.

iv) ~~fputc();~~

v) fgetc();

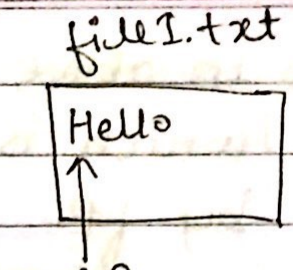
This function is used to read single character from file pointed by fp.

Syntax:

~~ch=fgetc~~ ch=fgetc(fp);

Example:

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



→ fgetc() function reads single character from file & stores in ch variable.

vi) fputc();

This function is used to store single character into file pointed by fp.

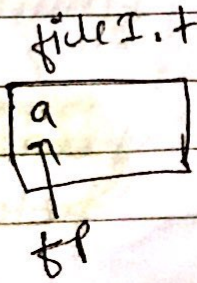
Syntax:

```
fputc(ch, fp);
```

→ ch is a variable that contains single character, which is stored in a file pointed by fp.

Example:

```
char ch='a';
fputc(ch, fp);
```



→ In above example ch has value "a" which is stored in a file pointed by fp.

2] write a c program to read contents of a file say abc.txt and count number of characters, number of lines, number of white spaces (blanks) & number of tabs.

```
#include<stdio.h>
void main()
{
    FILE *fp;
    char ch;

    int charcount=0;
    int linecount=0;
    int whitecount=0;
    int tabcount=0;

    fp=fopen("abc.txt","r");

    if(fp==NULL)
    {
        printf("file not found");
    exit(0);
```

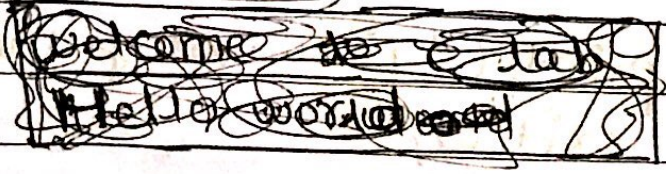
```
while((ch=getc(fp))!=EOF)
{
    charcount++;
    if(ch==' ')
    whitecount++;
    if(ch=='\n')
      linecount++;
    if(ch=='\t')
      tabcount++;
}
fclose(fp);

printf("character count=%d", charcount);
printf("whitespace count=%d", whitecount);
printf("line count=%d", linecount);
printf("tab count=%d", tabcount);
}
```

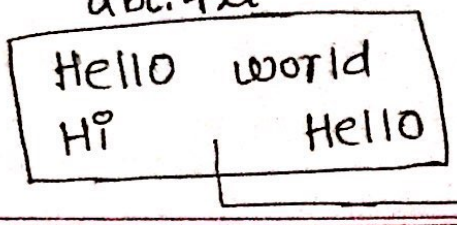
output:

Suppose abc.txt has following contents then output is shown as below

abc.txt



abc.txt



character count = 21
whitespace count = 1
line count = 2
tab count = 1

2] write a c program to copy contents of one file into another file.

```
#include<stdio.h>
#include<process.h>
void main()
{
   FILE *fp1, *fp2;
   char ch;
   fp1 = fopen("abc.txt", "r");
   if(fp1==NULL)
   {
      printf("file not found");
      exit(0);
   }
   fp2=fopen("file1.txt", "w");
```

~~white~~

P.T.O

```
while((ch=fgetc(fp1))!=EOF)
{
    fputc(ch, fp2);
}
fclose(fp1);
fclose(fp2);

fp2=fopen("file1.txt", "r");
while((ch=fgetc(fp2))!=EOF)
{
    printf("%c", ch);
}
fclose(fp2);
}
```