# ASSIGNMENT ON FILE

Mode : r

20/20

Group :
Roll%12 == 2

Members :
1805062
1805074
1805086
1805098
1805110

Introduction :
In this assignment , different functions related to FILE operations were tried out in various ways to see if there were any inconsistency/discrepancy. So far, everything we tried gave us the expected output. Nothing out of the ordinary happened.

## 1.READING A TEXT FILE USING “FREAD” :

We tried fread on a file where the following was written in text mode:
1234

The code is as follows :

```
    FILE *f1;
    f1 = fopen("test.txt","r");
    int a[100]={0};
    fread(a,sizeof(int),5, f1);

    int i;
    for(i=0;i<5;i++)
    {
        printf("%d ",a[i]);
    }
```

Which gave the output :

```
875770417 0 0 0 0
```

The number 875770417 was the 4 byte ascii value of 4 characters in the file (verified by using the online ascii to integer converter). Therefore fread is working how it was supposed to work.

When the input was changed to 12345 , the output was :

```
875770417 53 0 0 0
```

Which also matches the expected data.

Fread also works properly in the case of reading it character by character. Like :

```
    fread(str1,sizeof(char),100, f1);
    str1[26]='\0';
    printf("Str1 = %s\n", str1);
```

**Txt file had :** **ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**Output :** **ABCDEFGHIJKLMNOPQRSTUVWXYZ**

The only thing you have to do is after using fread to put the characters into an array, put a null character at the end of the string. Otherwise, the output would be something like :

**ABCDEFGHIJKLMNOPQRSTUVWXYZΓτï╚■╨Ç @**

The garbage values in the end is because the string wasn't terminated

## 2. USING FWRITE IN THE FILE:

Fwrite is intended to write into the file in binary. Needless to say it didn't work properly when the file was accessed in read mode.

## 3. USING FREAD TO READ FROM A BINARY FILE :

This also works, can output the intended result in the console even though the txt file was not readable by humans. Which was expected.

## 4. USING FSCANF IN THE TEXT FILE :

First we tried writing a number and then scanning it as integer.

Txt file had :    12345

**Code :**

```
int a[100]={0};
fscanf(f1,"%d",&a[0]);
int i;
    for(i=0;i<5;i++)
    {
        //fscanf(f1,"%d", &a[i]);
        printf("%d ",a[i]);
    }
```

**Output :**

```
12345 0 0 0 0
```

Success. No problem.

Then a txt file with "ABCDEFGHIJKLMNOPQRSTUVWXYZ" was scanned as an **integer(only the first byte).**

Code:

```
int a[100]={0};
fscanf(f1,"%d",&a[0]);
int i;
    for(i=0;i<5;i++)
    {
        printf("%d ",a[i]);
    }
```

The value of a[0] was not assigned. Output was :

```
0 0 0 0 0
```

My impression was that the output should have been : **65 0 0 0 0**
But I was wrong.
Scanf behaves like this as well (which apparently was new information to me). It doesn't take a character and assign it to an integer variable. If the integer was initialized then the output becomes the same as the initial value. If the integer was not initialized then a garbage value gets printed.

## 5. FSEEK TO SOMEWHERE IN THE MIDDLE OF THE FILE AND THEN PRINT USING FEOF AND FGETC AND USING FTELL TO PRINT WHICH BYTE IS BEING READ:

It should print the data from the location in the file fseek was pointing to. And **it did**.

**Code:**

```
    fseek(f1,5,SEEK_SET);
    while(!feof(f1))
    {
        ch = fgetc(f1);
        printf("%c",ch);
    }
    printf("\n%ld",ftell(f1));
    fseek(f1,5,SEEK_SET);
    rewind(f1);
    printf("\n%ld",ftell(f1));
    ch = fgetc(f1);
    printf("\n%c",ch);
```

The file was the same as before with all the letters of the alphabet.

**Output :**

```
FGHIJKLMNOPQRSTUVWXYZ
26
0
A
```

Conclusion, this works perfectly as well.

## 6. CAN I PRINT THE FILE IN REVERSE ORDER? DOES NEGATIVE OFFSET VALUE WORK IN FSEEK?

Short answer is yes I can, and yes it does.

Code :

```
    fseek(f1,0,SEEK_END);
    int size = ftell(f1);
    fseek(f1,-1,SEEK_END);
    printf("%ld\n",ftell(f1));
    int i;
    for(i=0;i<size;i++)
    {
        ch = fgetc(f1);
        printf("%c",ch);
        fseek(f1,-2,SEEK_CUR);
    }
```

Output :

```
25
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

In conclusion, I did not find any unusual behaviorregarding files and its functions. Every unexpected result could be explained.

[only one student contributed to this assignment]