Ch-10- File I/O

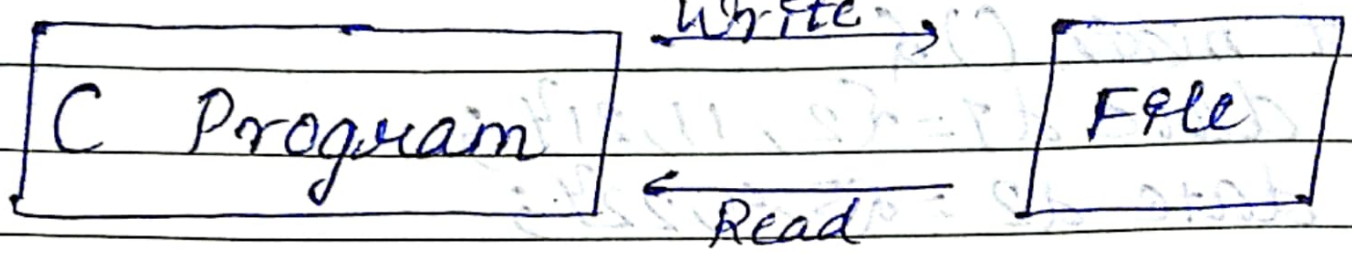
The Random Access memory is volatile & its content is lost once the program terminates. In order to persist the data forever we use files.

A file is data stored in a storage device. A C program can talk to the file by reading content from it & writing content to it.

→ When the mode is writing a file with the specified name is created if the file does not exist.
The contents are deleted if the file already exists

→ When the purpose is appending the file is opened with current contents safe. A file with the specified name is created if the file doesn't exist.

⇒ When the purpose is reading & if it exists, then the file is opened with current contents safe. otherwise error occurs.



FILE pointer

A 'file pointer' is a structure which needs to be created for opening the file.

A file pointer is a pointer to the structure of the file.

File pointer is needed for communication between the file & the program.

A file pointer can be created as follows:

```
FILE * ptr;
ptr = fopen("filename.txt", "mode");
```

File opening modes in C

C offers the programmers to select a mode for opening a file.

Following modes are primarily used in C file I/O

"r" → Open for reading → If the file does not exists, fopen returns NULL & contents are safe if it exists

"rb" → Open for reading in binary

"w" → Open for writing → If the file exists, then contents would be overwritten

"wb" → Open for writing in binary

r+ → The existing file is opened at the beginning for both reading & writing

w+ → The existing file is opened at the beginning for both reading & writing

a+ → The existing file is opened at the beginning for both reading & writing

"a" → open for append → If the file does not exist, it will be created & If it exists its current contents are safe.

Types of files

There are 2 types of files:

1) Text files (.txt, .c)
2) Binary files (.jpg, .dat)

Reading a file

A file can be opened for reading as follows:

```
FILE *ptr
ptr = fopen("Harry.txt", "r");
int num;
```

Let us assume that "Harry.txt" contains an integer. We can read that integer using:

```
fscanf(ptr, "%d", &num);
```

→ fscanf is file counterpart of scanf

This will read an integer from file in num variable.

Suppose we have a file called Harry.txt & we store in it

34 88

Code:

```
#include <stdio.h>

int main(){
    FILE* ptr;
    int num;
    int num2;
ptr = fopen("Harry.txt", "r");

fscanf(ptr, "%d", &num);

fscanf(ptr, "%d", &num2);

fclose(ptr);

printf("The value of num is %d\n", num);
printf("The value of num2 is %d\n", num2);

 return 0;

}
```

Output:
The value of num is 34
The value of num 2 is 88

Qii Quiz: Modify the above program to check whether the file exists or not before opening the file

```
#include <stdio.h>

int main()
```

```
FILE *ptr;
int num;
int num2;
ptr = fopen("Harry.txt", "r");

if (ptr == NULL){
    printf("The file does not exist\n");
}

else{
    fscanf(ptr, "%d", &num);
    fscanf(ptr, "%d", &num2);
    fclose(ptr);
    printf("The value of num is %d\n", num);
    printf("The value of num2 is %d\n", num2);
}

return 0;
}
```

Output
The file does not exist

Closing the file

It is very important to close the file after reading or writing. This is achieved by using fclose as follows:

```
fclose(ptr);
```

This will tell the compiler that we are done working with the file & the associated resources could be freed.

Writing to a file

We can write to a file in a very similar manner like we read the file

~~FILE *fpt~~

FILE *fptr

fptr = fopen("Harry.txt", "w");

Code:

```
#include <stdio.h>

int main(){
    FILE* fptr;
    int number = 45;
    fptr = fopen("generated.txt", "w");

    fprintf(fptr, "The number is %d", number);
    fprintf(fptr, "Thanks for using fprintf", number);
    fclose(fptr);

    return 0;
}
```

generated.txt file

The number is 45
Thanks for using fprintf

file: getcdemg.txt

this is my getc demo file

Code: #include <stdio.h>

int main(){
FILE *ptr
ptr = fopen("getcdemg.txt", "r");
//char c = fgetc(ptr)

~~printf("The value of my character is %c~~

printf("The value of my character is %c\n", fgetc(ptr));

//

return 0;

}

Output: The value of my character is t

//

h

i

s

space

fgetc() & fputc()

fgetc & fputc are used to read & write a character from/to a file.

fgetc(ptr) ⇒ Used to read a character from file

fputc('c', ptr); ⇒ Used to write character 'c' to the file

~~EOF: End of File~~

Code: #include<stdio.h>

```
int main(){
    FILE *ptr;
    ptr = fopen("putcdemo.txt", "w");
    fputc('c', ptr);
    {
    "
    return 0;
```

Output

~~ptl put c~~ putcdemo.txt

ccc

Code: # include <stdio.h>

```
int main() {
    FILE *ptr;
    char c;
    ptr = fopen("getcdemo.txt", "r");
    c = fgetc(ptr);

    while(c != EOF) {
        printf("%c", c);
        c = fgetc(ptr);
    }

    return 0;
}
```

Output

This is my getc demo file

EOF: End of File

fgetc returns EOF when all the characters from a file have been read. So we can write and check like below to detect end of file.

```
while(1) {
    ch = fgetc(ptr);
    if (ch == EOF) {
        break;
    }
    // code
}
```

⇒ When all the content of a file has been read break the loop!

fwrite function

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)

ptr - This is a pointer to array of elements to be written

size - This is the size in bytes of each element to be written

nmemb - This is the number of elements, each one with a size of size bytes

stream - This is the pointer to a file object that specifies an o/p stream

size_t fread(void *ptr, size_t size,
size_t nmemb, FILE *stream)

fread & fwrite are used for reading & writing to a binary file.

ftell() gives current pos^n in the file. (in terms of byte from start)

```
n = ftell(fp);
```

ftell takes a file pointer & returns a number of type long, that corresponds to the current position. This function is useful in saving the current pos^n of a file, which can be used later in the program.
n would give relative offset in bytes (or bytes) of the current position. This means n bytes have already been written or read.

fseek() sets the position to desired point in the file

```
fseek(file_ptr, offset, position);
```

file_ptr is a pointer to the file concerned, offset is a number or variable of type long, and position is an integer number. The offset specifies the number of pos^n s (bytes) to be moved from the location specified by position. The pos^n can take one of the 3 values.

Value	Meaning
0	Beginning of the file
1	Current position
2	End of file

If Offset is +ve, it means to move forwards
If " " -ve, - " " backwards

For e.g.-

```
fseek(fp, 0L, 0);
```
Go to the beginning (similar to rewind)

```
fseek(fp, 0L, 2);
```
Go to the end of file, past the last characters of the file.

```
fseek(fp, m, 0);
```
Move to $(m+1)^{th}$ byte in file

```
fseek(fp, -m, 2);
```
Go backward by m bytes from end.

Ch-10 Practice Set

Q1) Write a program to read 3 integers from a file

File: pr01.txt

23 54 67

Code:
```
#include<stdio.h>

int main(){
 int a, b, c;
 FILE *ptr;
 ptr=fopen("pr01.txt", "r");
 fscanf(ptr,"%d %d %d", &a,&b,&c);
 printf("The values of a b & c is
        %d %d %d\n", a, b, c);

 return 0;
}
```

Output
The values of a b & c is 23 54 67

Q2 Write a program to generate multiplication table of a given number in text format. Make sure that the file is readable & well formatted.

```
#include <stdio.h>

int main() {
    FILE *ptr;
    int num;
    printf("Enter the integer you need the table of \n");
    scanf("%d", &num);
    ptr = fopen("table.txt", "w");
    for(int i=0; i<10; i++) {
        fprintf(ptr, "%d X %d = %d \n", num, i+1,
                                        num*(i+1));
    }
    fclose(ptr);
    return 0;
}
```

Output

```
Enter the integer you need the table of
7
```

In file table.txt

```
7 X 1 = 7
7 X 2 = 14
   :
7 X 10 = 70
```

Q3) Write a program to read a text file character by character & write its content twice in a seperate file

File: a.txt

for problem number 3

Code:

```
#include <stdio.h>

int main(){
    FILE *ptr1;
    FILE *ptr2;
ptr1 = fopen("a.txt", "r");
ptr2 = fopen("b.txt", "w");

char c = fgetc(ptr1);
while(c != EOF){
  fputc(c, ptr2);
  fputc(c, ptr2);
  c = fgetc(ptr1);
}

fclose(ptr1);
fclose(ptr2);

return 0;
}
```

Output

File . b. txt

ffoorr ppprroo bb lleem ~~nnuummbb~~

nnuummbbeerr 33

Q4 Take name & salary of 2 employees as input from the uscs & write them to a text file in the following format:

name 1 , 3300

name 2 , 7700