Credits: Following online resources were used in preparing contents presented here

1. tutorialspoint.com
2. javapoint.com
3. geeksforgeeks.org

Some of the example programs have been modified by me to avoid confusions

File Handling in C

In programming, we may require some specific input data to be generated several numbers of times. Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

However, if we need to do so, we may store it onto the local file system which is non volatile and can be accessed any time. This can be achieved by using file handling features in C.

A file represents a sequence of bytes, regardless of it being a text file or a binary file. C programming language provides access on high level functions as well as low level (OS level) calls to handle file on your storage devices.

File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program. The following operations can be performed on a file.

- Creation of the new file
- Opening an existing file
- Reading from the file
- Writing to the file
- Deleting the file

Opening Files

You can use the `fopen( )` function to create a new file or to open an existing file. This call will initialize an object of the type `FILE`, which contains all the information necessary to control the stream. The `fopen()` function works in the following way.

- Firstly, It searches the file to be opened.
- Then, it loads the file from the disk and place it into the buffer. The buffer is used to provide efficiency for the read operations.
- It sets up a character pointer which points to the first character of the file.

The prototype of this function call is as follows:

```
FILE *fopen( const char * filename, const char * mode );
```

Here, `filename` is a string literal, which you will use to name your file, and access mode can have one of the following values:

Sr.No.	Mode	Description
1	r	Opens an existing text file for reading purpose.
2	w	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
3	a	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
4	r+	Opens a text file for both reading and writing.
5	w+	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
6	a+	Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

To handle binary files, following access modes are used instead of the ones mentioned above

```
"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"
```

Closing a File

To close a file, use the `fclose( )` function. The prototype of this function is:

```
int fclose( FILE *fp );
```

The `fclose( )` function returns zero on success, or `EOF` if there is an error in closing the file. This function actually flushes any data still pending in the buffer to the file, closes the file, and releases any memory used for the file. The `EOF` is a constant defined in the header file `stdio.h`.

C standard library provides various functions to read and write a file, character by character, or in the form of a fixed length string.

Writing a File

`fputc( )` is the simplest function to write individual characters to a stream.

```
#include<stdio.h>
int main()
{
    int i = 0;
        char filename[]="output.txt";
        char filemode[]="w";

    FILE *fp = fopen(filename,filemode);
    if (fp == NULL)
      return 0;
    char string[] = "\n Department of Computer Science \n & Information Technology \n";
    for (i = 0; string[i]!='\0'; i++)
        fputc(string[i], fp);
    fclose(fp);
        printf("\n string successfully written to file %s \n", filename);
    return 0;
}
```

The function `fputs()` writes the string `s` to the output stream referenced by `fp`. It returns a non-negative value on success, otherwise `EOF` is returned in case of any error. You can use `int fprintf(FILE *fp,const char *format, ...)` function as well to write a string into a file. Try the following example.

```
#include <stdio.h>
int main() {
   FILE *fp;
   fp = fopen("test.txt", "w+");
   fprintf(fp, "This is testing for fprintf...\n");
   fputs("This is testing for fputs...\n", fp);
   fclose(fp);
     return 0;
}
```

When the code in listing-2 is compiled and executed, it creates a new file `test.txt` and writes two lines using two different functions.

Reading a File

`fgetc` is the simplest function to read a single character from a file. The `fgetc()` function reads a character from the input file referenced by `fp`. The return value is the character read, or in case of any error, it returns `EOF`.

```c
#include<stdio.h>
int main( )
{
    FILE *fp ;
    char ch ;
    fp = fopen("test.txt","r") ;
    while ( 1 )
    {
        ch = fgetc ( fp ) ;
        if ( ch == EOF )
            break ;
        printf("%c",ch) ;
    }
    fclose (fp);
    return 0;
}
```

When the code in listing-3 is compiled and executed, it reads the file created in the previous section and produces the following result

The function `fgets` allows to read a string from a stream. It reads up to n-1 characters from the input stream referenced by `fp`. It copies the read string into the buffer `buf`, appending a null character to terminate the string.

If this function encounters a newline character `\n` or the end of the file `EOF` before they have read the maximum number of characters, then it returns only the characters read up to that point including the new line character. See listing-4

```c
#include<stdio.h>
int main(){
    FILE *fp;
    char text[300];

    fp=fopen("test.txt","r");
    printf("%s",fgets(text,200,fp));

    fclose(fp);
    return 0;
}
```

Deleting a File

The C library function `int remove(const char *filename)` deletes the given filename so that it is no longer accessible.

See listing-5 for example program.

```c
#include <stdio.h>
```

```c
int main ()
{
   int ret;
   char filename[] = "test1.txt";
   ret = remove(filename);
   if(ret == 0)
     {
      printf("\nFile deleted successfully\n");
   }
     else
     {
      printf("\nError: unable to delete the file\n");
   }
   return 0;
}
```