C Programming

File Input/Output (I/O)

Binary and ASCII - American Standard Code for Information Interchange (text) files

There are two types of files in C: text (ASCII) and binary. The difference between the two types of files is in the way they store numeric type (int, float, etc.,). In binary files, numeric data is stored in binary, while in text files numeric data is stored as ASCII characters

e.g.,

short num = 123;

This would occupy 2 bytes of memory in RAM.

Character	'1'	'2'	'3'
ASCII value in decimal	49	50	51
ASCII value in binary	00110001	00110010	00110011

Each digit of the number requires one byte of storage in an ASCII (text) file.

In a binary file, the digits of a number do not occupy individual storage. Instead, the number is stored in its entirety as a binary number.

Opening and Closing files

Before you can access a file for input (reading) or output (writing), you must first **open** the file. To open a file in C, you need to use the function **fopen()**

Let's have a look at a short example:

```
/*
```

```
Opening and Closing a file in C
*/

#include <stdio.h>

int main()
{
   //Create a file pointer to a file
   FILE *fp;

   //Open the file called file.txt for reading
   fp = fopen("file.txt", "r");

   // Check if the file was opened successfully
   if(fp == NULL)
   {
       printf("\nError opening file");
   } // end if
   else
   {
       printf("\nfile.txt opened successfully");

       // Close the file after completing all associated work
       fclose(fp);
   } // end else

   return 0;

} // end main()
```

Repl 24.1: https://replit.com/@michaelTUDublin/241-Open-Close-a-file#main.c

In the above example, a file called "file.txt" was opened for reading. The file must pre-exist before running the program and be located in the same working directory where the program is running.

Let's have a look at the other modes for interacting with a file.

Mode	Action
"r"	Open a text file for reading only. The file **must already exist** or else the file pointer is NULL.
"w"	Open a text file for writing only. The file is created if it does not already exist. **However, an existing file is replaced.**
"a"	Open a text file for writing. Data is **appended** to the end of the file, or a new file is created if the file does not already exist.
"r+"	Open a text file for **both** reading and writing. The file **must already exist** or else the file pointer is NULL.
"w+"	Open a text file for **both** writing and reading. The file is created if it does not already exist. **However, an existing file is replaced.**
"a+"	Open a text file for **both** reading and appending. Data is written to the end of the file, or a new file is created if it does not already exist.

If you wish to use a Binary file, you simply place the character 'b' to the end of the mode listed in the above table. For example, "rb", "w+b", "a+b", etc.,

In summary, the general format of **fopen()** is:

```
file_pointer  =  fopen( "filename",  "mode" );
```

Note: it is quite common practice to combine the opening of a file and check if this was successful in a single line of code. For example,

```
…

…

int main()
{
    //Create a file pointer to a file
    FILE *fp;

    //Open the file called file.txt for reading
    // and check if this is successful
    if ( (fp = fopen("file.txt", "a+")) == NULL )
    {
        printf("\nError opening file");
    } // end if
    else
    {
        printf("\nfile.txt opened successfully");
...
...
```

Reading a character from a file

The standard function **fgetc()** reads a single character from a file. It is the equivalent of getchar() that we previously used.

The format of fgetc() is:

char_in = fgetc(file_pointer) ;

This function reads a single character from the file opened with the file pointer. The function places the character read from the file in the variable `char_in`.

The contents of `char_in` becomes **EOF** if there is an error or the end of the file has been reached. EOF is defined as -1.

Let's have a look at a sample program using fopen():

```c
/*
Reading a character from a file
*/
#include <stdio.h>
int main()
{
   //Create a file pointer to a file
   FILE *fp;

   char char_in;


   //Open the file called file.txt for reading
   // and check if this is successful
   if ( (fp = fopen("file.txt", "r")) == NULL )
   {
       printf("\nError opening file");
   } // end if
   else
   {
       printf("\nFile opened successfully\n\n");

       // read each character separately from the file
       // and display to standard output
       while( ( char_in = fgetc(fp) ) != EOF )
       {
           printf("%c", char_in);

       } // end while

       // Close the file after completing all associated work
       fclose(fp);

   } // end else
   return 0;

} // end main()
```

Repl 24.2: https://replit.com/@michaelTUDublin/242-Read-a-character-from-a-file

Writing a character to a file

The standard function **fputc()** writes a single character to a file. It is the equivalent of putchar() that we previously used.

The format of fgetc() is:

fputc(char_out, file_pointer) ;

`fputc()` writes a single character, i.e., `char_out` to the file opened for writing with the file pointer `file_pointer`.

Let's have a look at a sample program using fputc():

```
/*
Program to copy a file character by character
*/
#include <stdio.h>

int main()
{
   //Create 2 file pointers, one to each file
   FILE *fp_in, *fp_out;

   char char_in;

   //Open the file called file.txt for reading
   // and check if this is successful
```

```
    if ( (fp_in = fopen("file.txt", "r")) == NULL )
    {
        printf("\nError opening file");
    } // end if

    // Open the file called new.txt for writing. This is
    // destination file being copied to. Check if this is
    // successful
    else if ( (fp_out = fopen("new.txt", "w") ) != NULL )
    {
        // copy the file from file.txt into new.txt
        // character by character
        while( (char_in = fgetc(fp_in) ) != EOF )
        {
            // write the character char_in to new.txt
            fputc(char_in, fp_out);

        } // end while

        printf("\n\nCopying completed successfully");

        // Close both files after completing all associated work
        fclose(fp_in);
        fclose(fp_out);

    } // end else if
    else
    {
        printf("\nError creating new file");
    } // end else

    return 0;

} // end main()
```

Repl 24.3: https://replit.com/@michaelTUDublin/243-Write-a-character-to-a-file