French-Azerbaijani University

Data Structures & Algorithms

DSA — Signature Project

Group Members :

Saida Sideif-zade — Fahmin Guliyev — Gulshan Mustafayeva

January 4, 2020

1 Introduction

1.1 Overview

This report discusses the result of the work done in a simple watermarking of a BMP formatted image by getting a new image that hides secret text in morse code (which is only known to its author) at some specific position.

1.2 Approach

The project aims to complete the task above,besnstrdes that the goal is to manipulate the BMP file format and work with bytes. Since the computer has to store information about every single pixel in the image, the file size of a bitmap graphic is often quite large.
In short, Bitmap Images are binary files which contain a pixel by pixel description of a graphical image. The value of each pixels is stored in one or more bits of data as defined bits per pixel (bpp).

2 Methodologies

2.1 Project Background

The main purpose of the project and specifications about the program are the following:

1. Understand what is a file format.
2. Learn to deal with bytes and not integers.
3. Learn about how to use big little-endian values.
4. Learn how to open / write files.
5. To be able to write the program where we manipulate BMP images.

2.2 How it Works

- The pragma pack(1) was chosen to be used after doing some researches found as the most appropriate one to align the structure byte by byte, and to avoid the unexpected behavior of compiler.

  Reference to internet:
  #pragma pack instructs the compiler to pack structure members with particular alignment. Most compilers, when you declare a struct, will insert padding between members to ensure that they are aligned to appropriate addresses in memory (usually a multiple of the type's size).

- Two headers were defined (BMPHeader and DIBHeader) to manipulate the basic characteristics of a file.As there is no *byte* data type in C, for easier understanding, the unsigned char data type is converted using **typedef** and replaced with field name *byte*.A struct RGB was created to store color values corresponding to RGB colors.
- Help option is implemented with "help" function which provides the user with manual on how to properly execute the program.
- Some conditions were put to check whether the program was called properly (with the correct parameters).
- The options mentioned by the user should be proceeded and the parameters taken from the user should be converted to "good" types. Thus, the following changes were made:
- Position is given as string x,y. 2 components are seperated by comma inside the given string so by recognizing the comma the string is split into 2 parts one of them being the **x** position and another one being **y**.

  NOTE: If *-pos* option is not mentioned,the position is taken as 0,0(*top-left corner of the image*)
- Afterwards the message taken from the user is converted to lower-case letters. Since we use the morse encoding, there's no difference between lower and upper cases.
- In case user mentions **-date** option, the system call "date" is made and displayed with format *d-m-Y*

  NOTE: Above format was chosen due to different language conversion problems with respect to morse, it is better to have a format that only keeps digits to show the date.
- Then, after all options are processed, the new string *text* (consisting the message/date) that will be hidden on the image is created.
- Once the text is created, it is encoded with morse code by calling the function *lettertoMorse()*.
  The code of text is stored in **codeinmorse** string.

Working with Bytes

- While working with old file containing the original signature, the size of file was calculated in order to create needed size byte array called *buffer* to store all the information about **.bmp** file.Then everything as in the original file was copied to *buffer*.
  At this point, we are done with original file.
- Two headers (bmph & dibh) were declared. Using the *memmove()* function, required amount of blocks of memory were transferred from buffer to headers.
- After all bytes are saved in buffer, the bytes that represent the pixels should be modified.

  NOTE: As the file is represented in little-endians and as we work in big-endians, the pixels that we modify should be transferred in a **vice-versa** order (not RGB, but **BGR**).
- Regarding to the "**Hint 2**" provided in project description, the research was made and with respect to the following information obtained from wikipedia :

  "*The pixel array is a block of 32-bit DWORDs, that describes the image pixel by pixel. Usually pixels are stored "bottom-up", starting in the lower left corner, going from left to right, and then row by row from the bottom to the top of the image.*"

  The following formula were derived:

$$BytePosition = (Height - PosY) \times Width \times bpp + PosX \times bpp$$

  Where $PosX$ indicates the first component and $PosY$ indicates the second component of the given position, and *bpp* indicating bytes per pixel is equal to 3.

  NOTE: For some special cases, when the code to be hidden is too long to fit in the image we need to continue on the next line.But as image is described from bottom to top, every time we reach the right most position we move the byteposition(the one that indicates where to write in the image) with respect to this formula :

$$BytePosition = BytePosition - bpp \times (2 \times Width - 1)$$

- After all modifications are made to *buffer*, we can copy the buffer into the **new file**.

2.3 Instructions to operate the program.

To run and to execute the **watermark.c** :

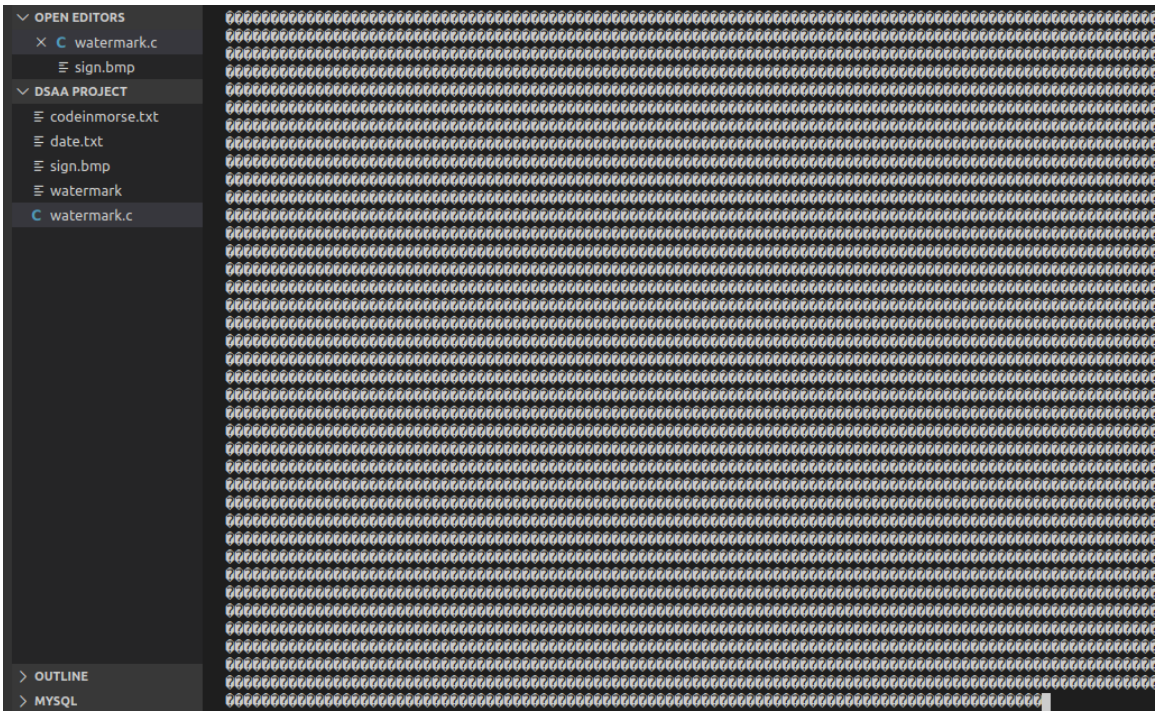
- gcc watermark.c -o wm
- ./wm sign.bmp **-text** "Hello" **-date -color** 243E7C **-pos** 10,20 -o modsign.bmp

To get some help executing the program, type :

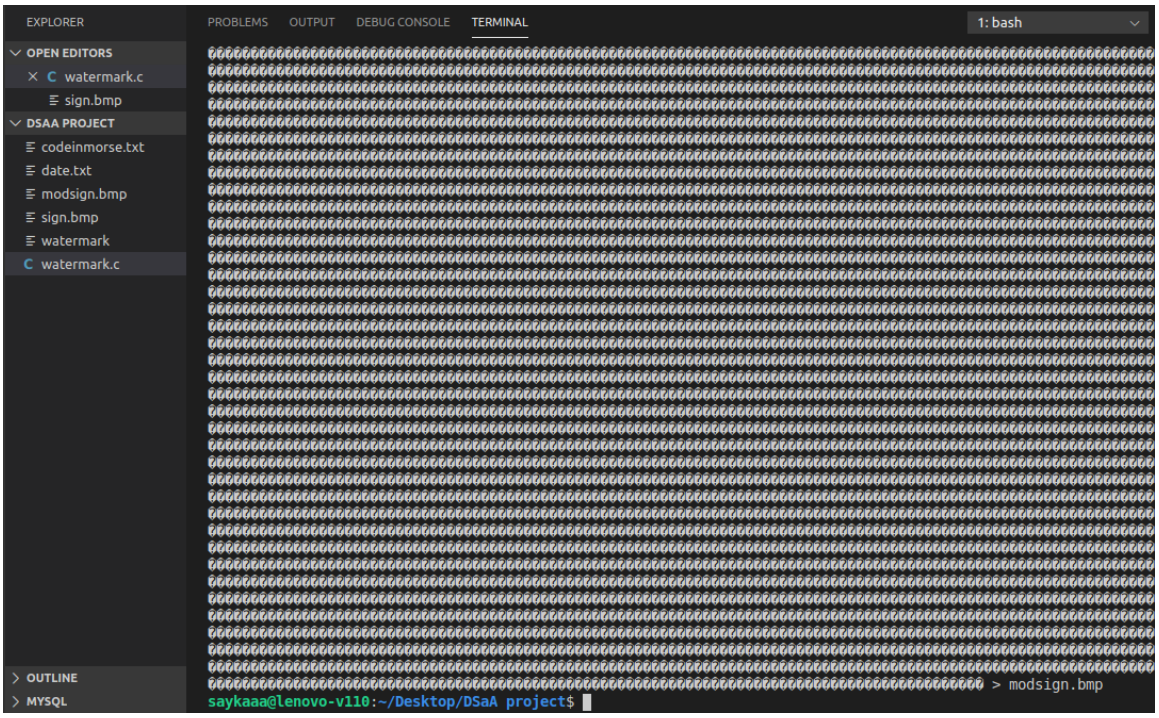
- ./wm **-h**
- ***NOTE :*** when **-o** option is not given while executing, the program will output on the standard output and it leads all the bytes to be seen on the screen(*stdout*).Eventually it forces the user to enter some output filename in which the new BMP image will be stored.(*the user will need to write* > ***modsign.bmp*** *to output in modsign.bmp*)
- After all bytes output to the standard output(**stdout**), it may be seen as a mistake at first, whereas it's not one.It's just way of implementation of redirecting to standard output.

The program was executed without the ***-o*** *option:*
./wm sign.bmp **-text** "Hello" **-date -color** 243E7C **-pos** 10,20

And the results were : (**BEFORE** and **AFTER**) the execution.



After the execution, the result:



3 Conclusion

In conclusion,we would like to say we studied the BMP file format to the certain extent for accomplishing the given task. We have learned how to deal with bytes and how to work with files. All these lead us to obtain the watermarked image with the desired secret message.

The program was then executed by the following command :
./wm sign.bmp **-text** "Hello" **-date -color** 243E7C **-pos** 10,20 **-o** modsign.bmp

References

[1] Ascii-Table — Your Web Reference : Morse Code,
Originally created for Samuel Morse's electric telegraph in the mid-1830s, it was also extensively used for early radio communication beginning in the 1890s.
http://ascii-table.com/morse-code.php

[2] Overleaf learn wiki — a wide range of help and information on Overleaf and LaTeX,
Overleaf Guide, LaTeX Basics,Mathematics,Figures & Tables,References & Citations etc.
https://www.overleaf.com/learn/latex/Tutorials

[3] StackOverFlow : Forum — Topic : "Read file byte by byte using C",
Some direction on reading file byte by byte in C,bunch of codes to work with to train.
https://stackoverflow.com/questions/51224682/read-file-byte-by-byte-using-c

[4] PRAGMA — Pragma Pack Effect in C,
Another StackOverFlow Forum Discussion concerning #pragma pack,explanation of #pragma pack preprocessor statement and why one would want to use it.
https://stackoverflow.com/questions/3318410/pragma-pack-effect

[5] GeeksForGeeks Tutorial — Memmove(); function in C/C++,
Short explanation about the function, a sample C program to show working of memmove(); and how it is different from memcpy();.
https://www.geeksforgeeks.org/memmove-in-cc/

[6] GeeksForGeeks Tutorial — Rewind(); function in C,
Short explanation about the function, a sample C program to show working of rewind(); and why fseek() would be preferred over rewind() in C.
https://www.geeksforgeeks.org/g-fact-82/

[7] GeeksForGeeks Tutorial — Date Command in Linux,
Date command in Linux with examples.Configuration, Preferences & Usage of the command.
https://www.geeksforgeeks.org/date-command-linux-examples/

[8] StackOverFlow: Forum — Discussion : "How to convert String to Hex Value in C",
Interesting Answer: this very question is often asked, but it's not quite the right question. Better would be: "How can I convert a hex string to an integer value?".
https://stackoverflow.com/questions/29547115/how-to-convert-string-to-hex-value-in-c/29547

[9] TechOnTheNet: C Library Functions, stdlib.h Library, strtol() function,
Convert String to Long integer,function usage, parameters and return type.
https://www.techonthenet.com/c_language/standard_library_functions/stdlib.h/strtol.php

https://www.overleaf.com/read/ddcmhzbksydw