Elie Daou

Assignment #1

Due Tuesday 7, February 2017

The objective of this assignment was four parts. The first parts were to read and write hexadecimal digital image files. The second part was to learn how to reduce an image file in 2 different directions. The last was to use xv to view hexadecimal image files.

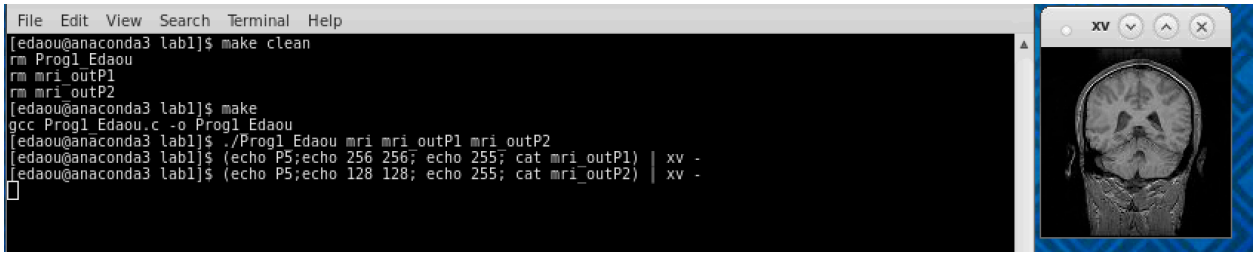
The background of the assignment is that images are stored as basic binary bit streams. We were supposed to read the bit stream and translate it into another image file. Also, we needed to use basic bash commands.

The only algorithm, per se, that was used was a nested for loops. 2-D arrays could have been used as well. There are two parts to this lab. The first part reads an input image (256x256) and outputs a (256x256) image file that is modified from the original image. The left half is left untouched. The second part reads an input image (256x256) and outputs a (128x128) image file that is the original image scaled down.

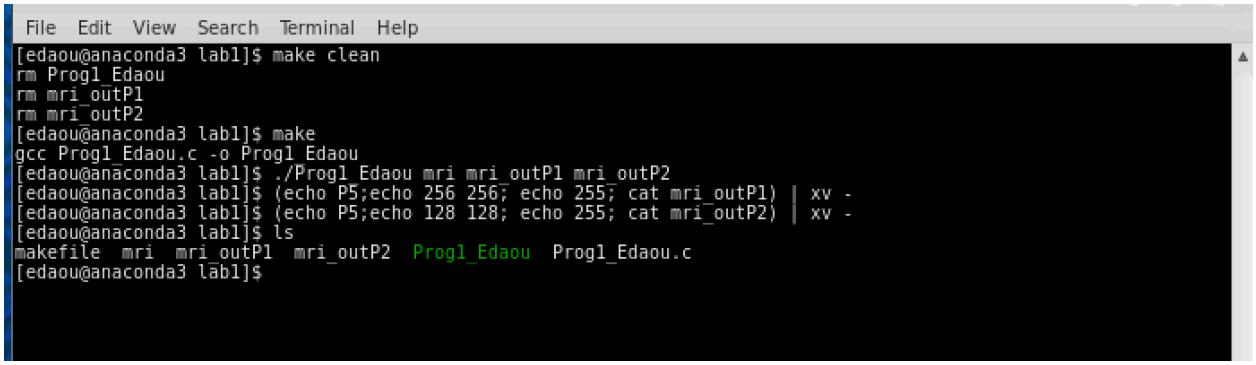
Results

Part 1

Part 2



Output Files



There were not many observations in this program. We noticed that if your iterations were off by one (1) pixel every iteration the loop, the whole picture ends up being distorted. Also, if, while using XViewer, the inputted xsize and ysize parameters are incorrect, the picture viewing ends up being distorted.

In conclusion, this lab was not terribly difficult but excited me to do more in-depth projects.

Source Code:

```
/* Program 1 - Elie Daou
 * Due (was) Wed 2 February 2017 9:00am
 * This program reads an image and outputs either:
 * - output1 will be the left half of a 256x256 image
 * - output2 will be the entire 256x256 image scaled down in half with an output of 128x128
 * */

#include <stdio.h>

int row;
int column;
unsigned char black = 0;
unsigned char pixel;

FILE *output1;
FILE *output2;
FILE *input;

int main(int argc, char *argv[]){
   //input image file
   input = fopen(argv[1], "rb");
   if (input == NULL){
```

```
    return 1;
  }

  // open both output image files
  // output1 will be mri_outP1 aka left half of the image
  // output2 will be the even pixels only (making it 128x128)
  output1 = fopen(argv[2], "wb");
  if (output1 == NULL){
    return 1;
  }
  output2 = fopen(argv[3], "wb");
  if (output2 == NULL){
     return 1;
  }

  // Read the image in a 2-d loop
  // First you go by column
  for (row = 0; row < 256; row++){
     // then by row
     for (column = 0; column<256; column++) {
        //always goingto be reading first, no matter what, so put it in
        fread(&pixel, sizeof(char), 1, input);
        //if in the left half of the picture, output it to output1
        if (column < 128) {
           fwrite(&pixel, sizeof(char), 1, output1);
        }
        // in the right half of the picture print black dots
        else{
           fwrite(&black, sizeof(char), 1, output1);
        }
        // if the row and the column is even, output it. otherwise, do nothing. it will be a smaller image.
        if ((row % 2 ==0) && (column % 2 == 0)) { //even column and even row
           fwrite(&pixel, sizeof(char), 1, output2);
        }
     }
  }

  fclose(output1);
  fclose(output2);
  fclose(input);
  return 0;
}
```