John O'Donnell
CSC 412 Operating Systems
Dr. Herve
11 October 2020

Programming Assignment 03 Report

This programming assignment required implementing a C program to match patterns in really simplified image files, and writing a bash script which calls this C program on every file in a given directory.

The design of my solution to this assignment started with designing a two dimensional array. The data structure is an abstraction layer between the user and a one dimensional array, so it's contents are easily used in two dimensional applications. The struct has a few private members defining its qualities, such as number of rows, number of columns, and the size of the data type the array will contain. This makes the array general and widely applicable: it isn't hard coded to expect character input, but instead can be instructed on initialization the which data type it is being made to contain. This data structure is defined and implemented in my files `array2d.h` and `array2d.c`, respectively.

My solution has another module, `fileOps.h` and `fileOps.c`, which defines and implements file operations for the program. These operations include opening files and creating a new two dimensional array from the file's data, and recursing through a directory to find the files of a particular type within.

The pattern searching algorithm I implemented is rather straightforward, and is outlined by the steps below. This algorithm is the one implemented in my C program, so it assumes the algorithm has been provided a single `.pat` file, a directory with `.img` files, and an output directory.

1. Fill a 2d array with the data from the given `.pat` file.
2. For each file in the directory, if the file extension is `.img`, add the file to a list.
3. For each file in the list:
    1. Create a new 2d array.
    2. Fill the array with the data from the `.img` file.
    3. Initialize a pattern match counter to 0.
    4. Allocate memory for a list, consisting of the match counter and match locations.
    5. For each element in the 2d array:

    1. Check if the pattern could fit at the index by comparing the height and width of the pattern array to the distance of the element from the bottom and right sides of its own array. If the pattern can not fit, no further checking is needed, and the patter does not match at this element.
    2. Assuming the pattern can fit at the current element, check each element of the pattern against it's counterpart in the image array. This iterates through the pattern in row-major order, comparing it to it's counterpart in the subarray of the image. If any of these comparisons fail, no further checking is needed, and the pattern does not match at this element.
    3. If every element in the pattern matches it's counterpart in the image subarray, then the program increments the pattern match counter by 1, and add the match locations to the list.
6. Output the match data to a file using the `fileOps.h` interface.
7. Free memory allocated for filenames, the match list, and the images's 2d array.
4. Free memory allocated for the list of files, file paths and names, and the pattern's 2d array.

I enjoyed this assignment thoroughly. I found it challenging, but not so challenging that I wasn't able to complete it. I had the most trouble with my implementation of the type-generic two dimensional array, which wasn't even necessary, so it was my own fault.

I have not found any inputs that break either my C program or my bash script. I have a good deal of type-checking and range confirmations, so that incorrect inputs or poorly formatted pattern or image files with be rejected by the program instead of crashing.