Korakot Santiudommongkol

CSC 412 Programming Assignment 04

<u>Tile.c</u>

The way I designed this operator first to get how many images should on each row and each column. My idea of making it “most square” is that there is the same amount of images on the row as the column. To get the number of images per row and height, I just get an infinite loop that adds one to an int numrow and numcol and then check if numcol*numrow is exactly the number of input images. If the number of input image is not a perfect square I’ll subtract one from int numcol and check if its equal or between before subtracting the value. The new height and width of the output image would be numcol*width, numrow*height. The way I create the output image is to take one of the input images one at a time and adding it to the output raster, one row at a time.

The only limitation for my tile operators is dealing with input images with different dimensions, I have only made my program with account for input image with all the same dimension. The other limitation is that the composite image would come out of a “most rectangle” image rather than “most square” image if it deals with input image where width!=height.

<u>Dispatcher.c</u>

The way my dispatcher is designed is to first argument check which check to make sure there are at the very least 4 arguments, my idea to get a complete argument check is by checking for all the commands in argv first. My program then check for number of arguments, after getting a number of argument I know how big my array needs to be to create one int array to hold the index of where each argument is stored in argv, then get a string array to store each argument which will correspond with the index array. After getting a complete command list I can fully check the number of arguments there needs to be. There are three different commands that takes different number of arguments, I have a function check argument to count how much of each command there are in the commands list. Then I multiple the number of each different command with the number of arguments needed to run the command, add all the values after multiplying it then add 3 which represents the first 3 commands in argv being the executable, input directory, and output directory. After checking for argument program goes on to dispatching all the commands. Dispatching functions creates child process equal to the number of commands there are and return an integer for the number of failed commands. The child process will execute the commands with the required arguments while the parent argument will wait for the child process to terminate, and get an exit status to see if failed or succeed. Exit status 0 means it is successful therefore, if the status is not 0 it ended as a failure and then add one to numfail int. The program will then exit with the number of failed process.

The limitation for my dispatch.c is mainly with the arguments, the only way I checked for arguments to see if the number of argument is correct, I did not check for any kind of mismatched arguments. My program would end as a segmentation fault if there is a command string at the very end of argv list and the number of required arguments is correct.

Bash script

Script is straightforward. Check for number of argument, check if path to input directory exists if not exit the script, and check if path to output directory exists if not create that directory. The script moves on to compiling the tile.c program correctly. Then I used an array to store all the .tga file found in the input directory, then sort using the sort function on bash to sort the list. To get the output image name used a loop to from 0 to 100 and check if a file with that name exists or not, if not use that string for the output, then run the executable with all required arguments. The only limitation to this script is if the input directory does not hold any tga file.