Command Line Arguments

Lab #3: Section J

Submitted By: Wen Zhanghao

Submission Date: March 26, 2015

Overview

Generally speaking, the main function always take no arguments, However, In this lab, new feature of C have been developed that main function take arguments argc and argv, which read the enter information from terminal. And we use this to apply into ppm image code from last lab.

Analysis

Function argc actually count the inputs after run the c file, separated by white space between string. Counting process include ./xxxx as the first string. argv store the strings as array.

```
./program first second third
```

Here, `argc` is 4 and `argv` looks like this:

	0	1	2	3	4	5	6	7	8	9
0	'.'	'/'	'p'	'r'	'o'	'g'	'r'	'a'	'm'	'\0'
1	'f'	'i'	'r'	's'	't'	'\0'				
2	's'	'e'	'c'	'o'	'n'	'd'	'\0'			
3	't'	'h'	'i'	'r'	'd'	'\0'				

It is not hard to find that argc, and argv are always count from the order of user's input order — left to right. We can locate each char by using argv[][] because argv is two-dimension array. if we express in this way: argv[], it would represent the string of corresponding order.

The lab manual cleary state that when we call the add operation which is 'a', the program will take the next three command line arguments. when we call the mean filter operation 'm', the program will take next

command line argument. In other words, when we figure out the position of a or m, we can use array of argv to determine arguments that after them.

The command line arguments are all char, but if we want to add value, we should change it to type int. function atoi () helps the conversion.

Design

Lets start from difference in main function compared last time. At first we need to get data from image. Well, before we doing some modifications on image, we have to had image first. So now we have image data. Because argv&argc would go through all command arguments, we use loop to do it. For loop has been used because &argc count how many arguments entered in.

Then we meet if function. The arguments in the command line are strings that made of char. ‘a’ and ‘m’ have two way to understand: first, they can be understand into two char that in the first position of their array(although the array only one element itself plus /0). The second way is to treat them as string. I chose the second way to compare ‘a’ in string way. So strcmp need to be used to compare two strings. strcmp(argv[i], “a”).
Note:1. compare string by using double quote, char by using single quote.
2. if two string compared in strcmp are equal, they would return 0, which in if conditions as condition fails. So use ! before strcmp to continue code if strings are equal.

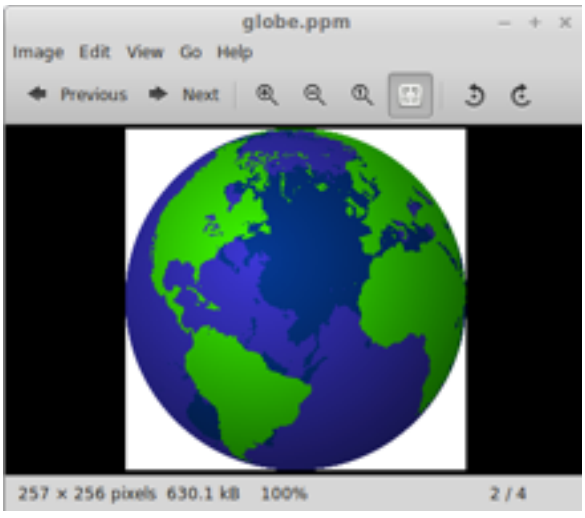
Then I used atoi at to pick up three int, adding them into red, green, blue separately. One thing should be avoided is that when the color value

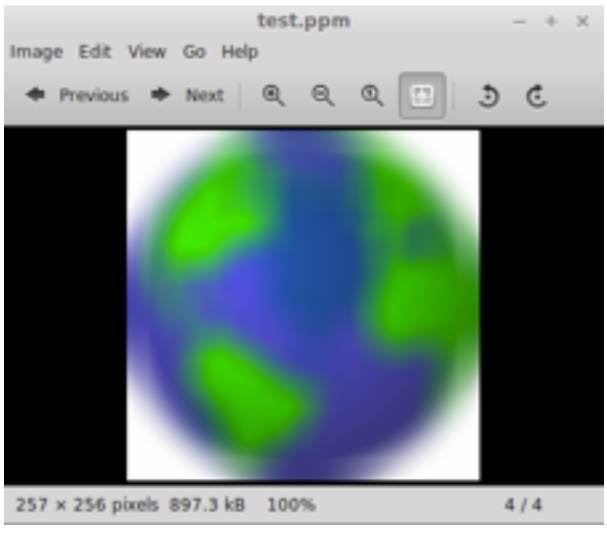
over number 255 after adding, we should change it into 255. Also, when we value drop below 0, we should change the color value into 0.

About mean filter is easier, we only need to get one value using atoi to convert string after char ‘m’ into int and apply this int as the argument width in mean function. In the main function, there are a source image and a destination image, and destination image is kind of the output if we run the meanf function, therefore, I used nested loop to cover original image with destination image. So in this way, the argument in ppmout does not need to be changed. It can keep argument ‘image’. Hence, whether or not command line argument include ‘m’ or not, it will be executed anyway.

Testing

Here is the image globe.ppm and the test.ppm image after run the program with the arguments “a 30 30 30 m 30”.



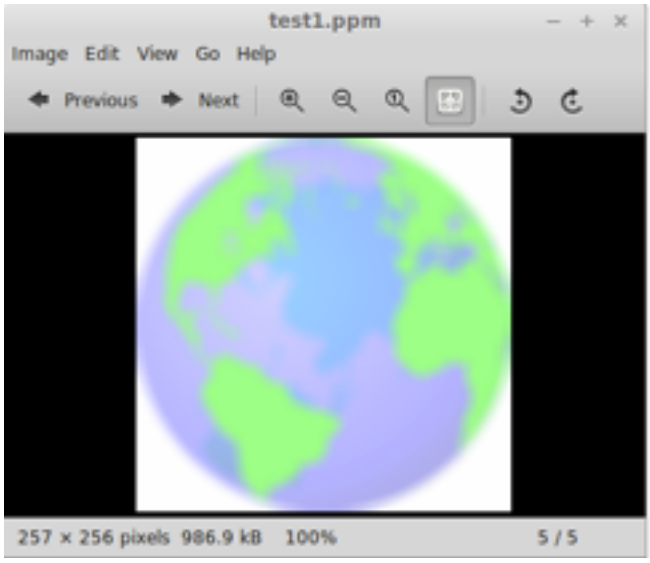


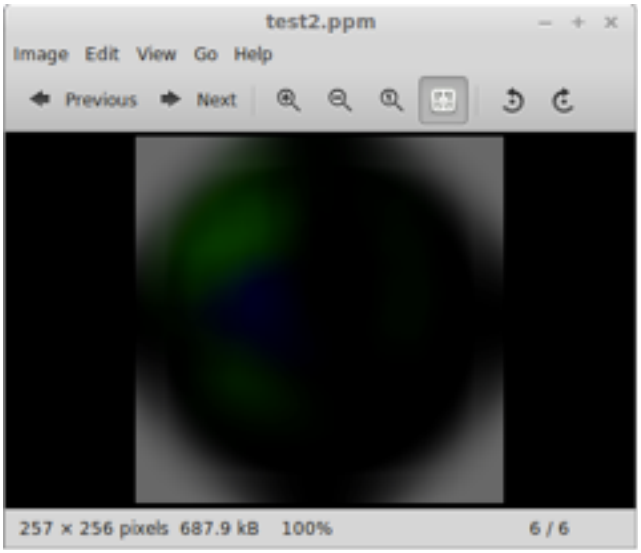
As we can see, the right image is kind of blur out and bright compared to the left image.

The two figures below are two other examples. The left one with a 150 150 150 m 10

the right one with a -150 -150 -150 m 40

So adding positive value makes image become brighter, and adding positive value makes image become more vague.





This code is so long, and formatting issue become more important. Overall I used three nested for loop and mixed with if function. There are a lot of brackets that we should be aware. I made one mistakes that I call ppmout in the for loop rather than the main function. When code is too long, like this time code in the main function are more then 60 lines, we have to scroll up and down to judge which bracket match another.

Extra Credit Part

In the last part, we run the code using argc, argv, features with lots of arguments. And then we redirect the input and output. If we do not redirect output, the terminal would display all the numbers. If redirect the output into another .ppm rather than showing on the terminal, the data would be transferred. In stead of the standard input and output, ppread and ppmout function have been changed that can read data from a file and output the data into another file.

fopen, fprintf, and fscanf

Comments

This lab includes lots of new knowledge. The requirement of lab manual is not very similar with the coed in the lecture. So we have to fully understand it before programming. In addition, as the course become more complicated, we need to do more research online and find out which new function and code should be used and how to use them.