

Lab 2

CPSC 1021- Spring 2017

**Due Date: Thursday, January 26, 11:59:59, midnight**

**Introduction**

Todays lab is designed to give you practice creating a ppm image. As well as reading in a ppm image and printing it back out.

**Lab Objectives**

* Practice using command-line arguments
* Practice using file redirection
* Practice using file pointers
* Practice using fprintf and fscanf
* Improve your programming skills
* Improve your problem solving and critical thinking skills

**Prior to Lab**

You may want to review your notes on creating a ppm image.

For all PPM headers in this course we will use P6 as the magic number. Why, as someone pointed out in class, and I totally misunderstood (sorry), a P6 image **is** **smaller in size**. The reason for this is P6 uses a ASCII decimal which is represented by a character which is 1 byte of data per color channel, whereas a P3 uses an integer which is 2 – 4 bytes of data per color channel.

You may use P3 if you need to debug. But when turning in your images you should use P6.

**Lab Instructions**

On the SoC server, change directory (cd) to your cpsc1021Labs directory (created last week in lab). Make a new folder (mkdir) called Lab2. Download any lab files from canvas and save them in your new Lab2 folder.

**Assignment**

**Part 1:**

For the first part of this lab, I will provide you with a ppm file. You will need to read in and then print the image back out. Make no changes to the image, simply read the image information from the file and print it back out. To simplify things everything will be written in main. Part 1 will use command-line arguments to determine the input file and the output file. The name of the program for part will be **lab2\_part1.c**. The input file will be named **part1\_input.ppm** and the output file should be named **part1\_output.ppm**.

Steps needed:

1. Declare a file pointer for the input file and one for the output file. Use command-line arguments to determine the names of the files to open. Google FILE\* i/o in c if you need a refresher on file pointers.
2. Make sure both files opened successfully.
3. As discussed in class there are two important parts of a ppm image – the header and the pixel values (the body). Remember, in class when we created an image we first wrote the header. Well, when you read an image you first must read the header. In this step read the header from the input file using fscanf. Then using fprintf print the header to the output file.
4. Next just as we did in class, use a loop to read in from the input file and print out to the output file, the pixel values.
5. View your images to make sure they are the same. If they are the same go on to Part 2 of this lab.

Compile and Execute

You should compile and execute this code in the following manner:

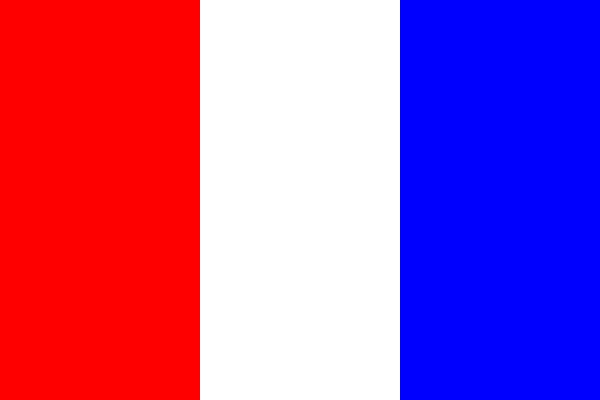
gcc lab2\_part1.c –o part1 -Wall

./part1 part1\_input.ppm part1\_output.ppm

**Part 2:**

You will create a second ‘c’ file for Part 2. You will name the program file **lab2\_part2.c** . This part will not utilize file pointers nor command line arguments. You will use redirection to determine the name of the output file, just as we did in class on Monday.

For the second part of this lab you will be allowed to use creativity in creating your image. Your image should be created procedurally. The image we created in class was created procedurally using only one color. I want you to create an image that utilizes two or more colors. As an example this is one I created using three colors:



Please don’t duplicate my image I want you to be creative. Have fun and create something interesting.

**You should name the output image in the following manner. It should be your username.ppm.** As an example my output file for part 2 would be yfeaste.ppm. **IF YOU DO NOT NAME THE OUTPUT CORRECTLY 10 POINTS WILL BE DEDUCTED.**

Compile and Execute

You should compile and execute this code in the following manner:

gcc lab2\_part2.c -o part2 -Wall

./part2 >yfeaste.ppm (remember your username should replace my username).

FORMATTING:

1. Your program should be well documented
2. Each file should include a header (example below)
3. Your program should consist of proper and consistent indention Ex. You should choose a number of spaces to indent – 3, 4, or 5 – and be consistent
4. No lines of code should be more than 80 characters

5 – 10 points will be deducted for each of the above formatting infractions. Below is an example of a program that meets each of the above formatting requirements:

/\*This is a simple program that prints hello world a specific

number of times. A ‘for loop’ is used to print the statement

10 times.\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

your name

username

Lab 1

Lab Section:

Name of TA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

int main()

{

int i = 0;

for( ; i < 10; i++)

{

printf(“hello world\n”);

}

return 0;

}

**Submission Instructions**

* Test your program on the School of Computing server prior to submitting.
* Use the tar utility to tar.gz all your files. You should include lab2\_part1.c part1\_input.ppm part1\_output.ppm, lab2\_part2.c and the ppm file for part 2. (see above for proper naming of the ppm file) 10 Points will be deducted if you do not name your files correctly.
* Name your tarred file Lab2.tar.gz
* Use handin(<http://handin.cs.clemson.edu)> to submit your Lab2.tar.gz file