# **Report: Image Inversion using libpng**

## **Introduction**

This report outlines a C program that utilizes the libpng library to read, invert, and write PNG images. The primary functionality of the program involves reading an input PNG image, inverting it vertically, and then writing the inverted image to a new PNG file.

## **Development Environment Setup**

### **1. Prerequisites**

* Ubuntu Environment: Ensure that you have a Linux environment, specifically Ubuntu, installed on your system.
* GCC Compiler: Install the GNU Compiler Collection (GCC) for compiling C programs.
* Make Tool: Install the make tool for managing the build process.
* libpng Library: Install the libpng library for PNG image processing.

sudo apt-get update

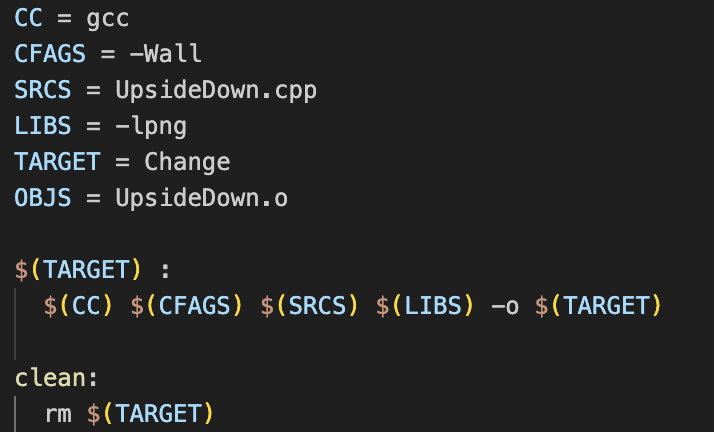
sudo apt-get install libpng-dev

### **2. Code Repository**

Clone the code repository from its source (e.g., GitHub).

In the code repository we have the source code “UpsideDown.cpp”, the make file “Makefile”, the image “Example.png” and the compile file “Change”.

Here’s is the content of the make file.



git clone <repository\_url>

cd <repository\_directory>

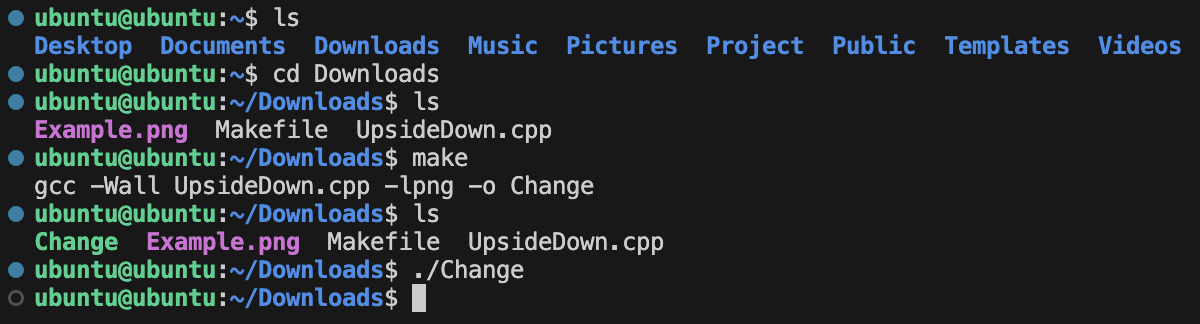
### **3. Set up the environment**

To set up the environment the user should first open ubuntu linux server with a virtual machine named UTM. After running the ubuntu server, we use the ssh server function of vscode to access the ubuntu server. To do that the user needs the ip address of the ubuntu server. The ip address is found by the command ifconfig on the machine’s terminal.

After selecting the ip address, the user logs in using their password.

From this point, the user can open the terminal and use the ls command to see all the folders in ubuntu server. After the user need to open the folder containing the code repository. They can dig in the folders until finding the write folder. After that, they can choose the folder make by typing the command “make”. The make file creates a executable file called Change. To see that file we can use ls again.

Finally, we can run using the command ./Change



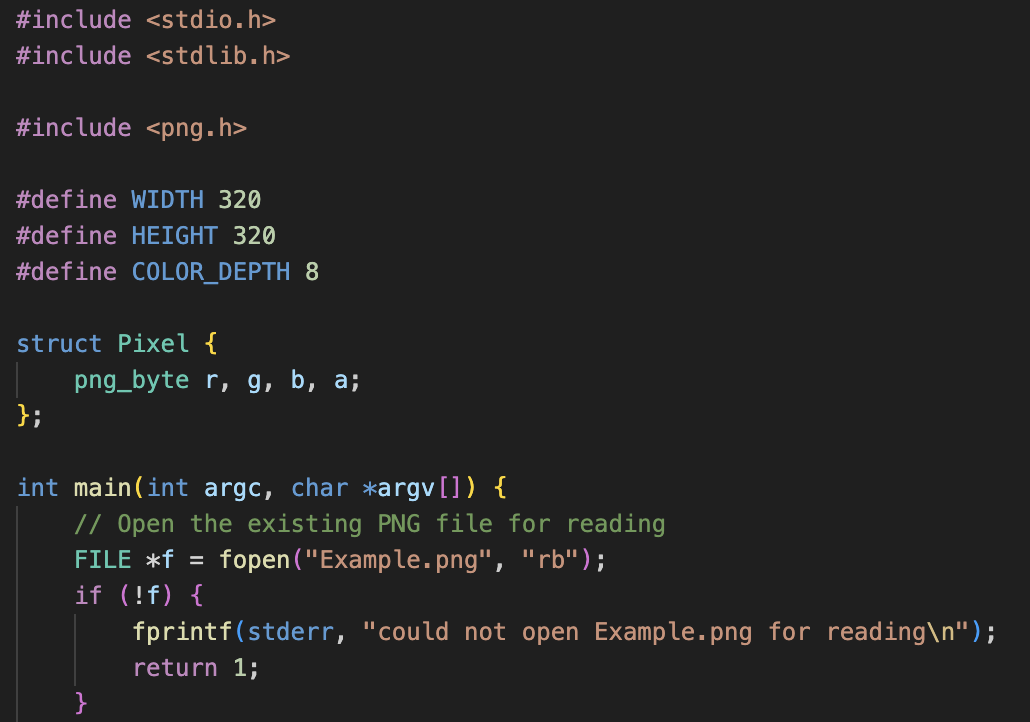
**Program Implementation**

### **Overview**

The program reads a PNG image, inverts it vertically, and then writes the inverted image to a new PNG file. It uses the libpng library for PNG image processing.

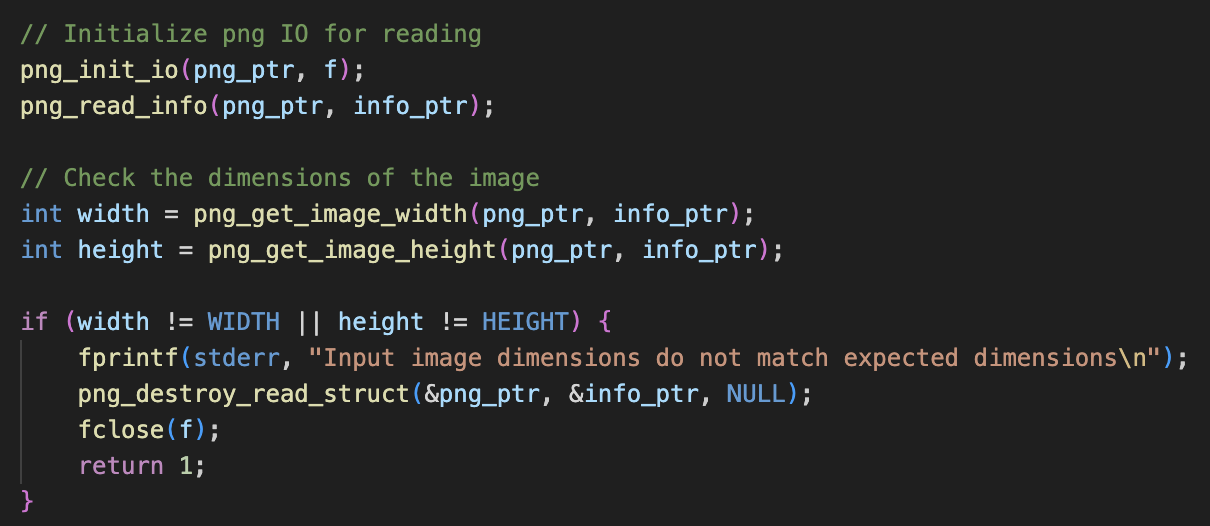
### **Code Explanation**

1. File Opening and libpng Initialization:



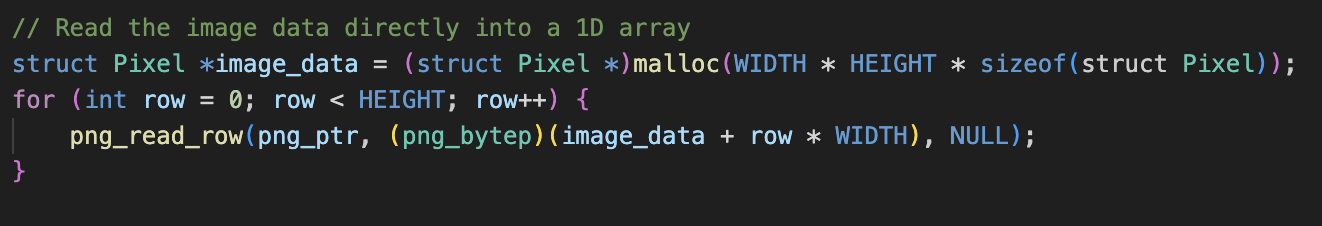
The code opens the PNG file for reading using fopen. Initializes libpng structures for reading using png\_create\_read\_struct. Sets up error handling using setjmp for early error detection during PNG reading. Initializes PNG I/O for reading using png\_init\_io and reads image information using png\_read\_info.

1. Image Dimension Check:



Checks if the dimensions of the input image match the expected dimensions .

1. Memory Allocation and Image Reading:



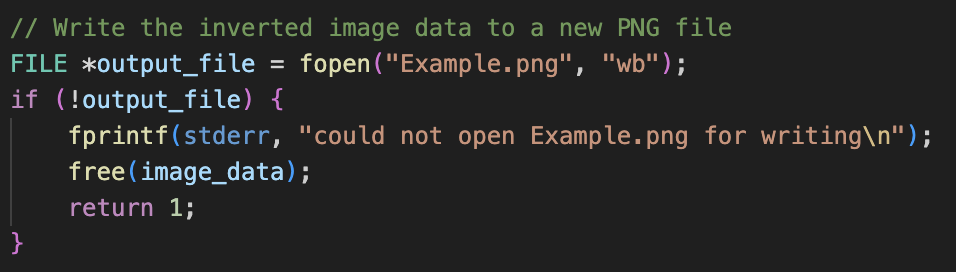
Allocates memory for storing image pixel data and reads the image data directly into a 1D array.

1. Invert Image Data:

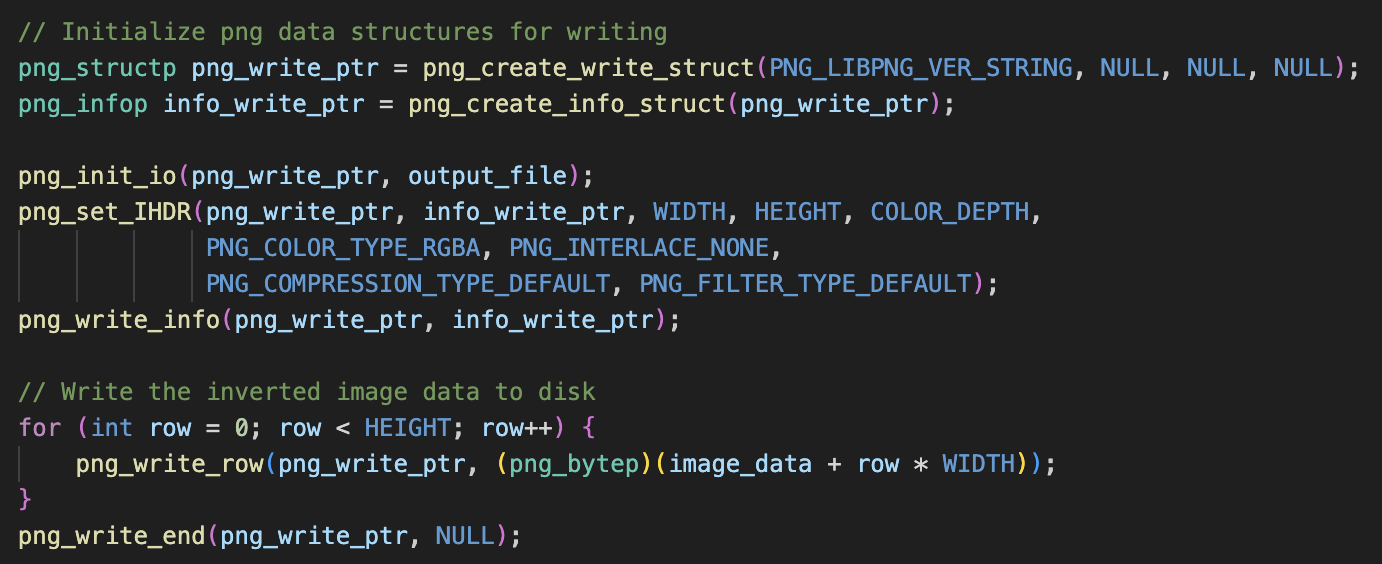


Inverts the image data in-place by swapping pixel values vertically.

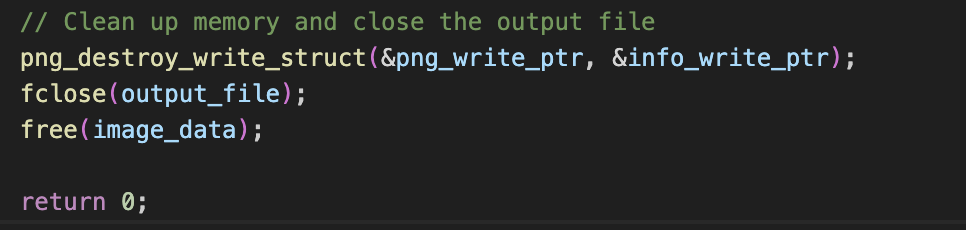
1. Write Inverted Image:



Opens a new PNG file for writing, initializes libpng structures for writing, and writes the inverted image data to the new PNG file.



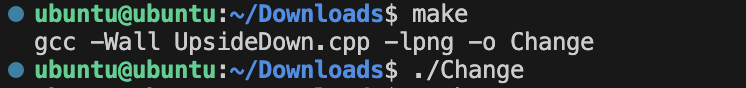
1. Clean Up:



Frees allocated memory and closes the input and output files. Cleans up libpng structures.

## **Running the Project**

To run the project we need to setup the environment and run the commands below.



## **Conclusion**

This program demonstrates a simple image inversion process using the libpng library. It reads an input PNG image, inverts it vertically, and writes the result to a new PNG file. The development environment setup ensures that the necessary tools and libraries are available for successful compilation and execution.

Note: Ensure that the input image ("Example.png") is present in the working directory or provide the correct path in the code. Adjustments may be needed based on the specifics of your environment.

Here is an example of the expected result :

Before:



After: