PPM Image Transformations

Learning Objectives

Upon completion of this assignment, you should be able:

1. To develop, compile, run and test C programs in a Linux environment
2. To navigate Linux command lines reliably

The mechanisms you will practice using include:

- Linux command lines: manual pages, Linux commands
- C Programming: structs, pointers, memory allocation, getopt

Program Specification

NAME

`ppmcvt` – convert ppm files

SYNOPSIS

`ppmcvt` [bg:i:r:smt:n:o:] file

DESCRIPTION

`ppmcvt` manipulates input Portable Pixel Map (PPM) files and outputs a new image based on its given options. Only one option that specifies a transformation can be used at a time.

In the synopsis, options followed by a ':' expect a subsequent parameter. The options are:

-b
: convert input file to a Portable Bitmap (PBM) file. (DEFAULT)

-g:
: convert input file to a Portable Gray Map (PGM) file using the specified max grayscale pixel value [1-65535].

-i:
: isolate the specified RGB channel. Valid channels are "red", "green", or "blue".

-r:
: remove the specified RGB channel. Valid channels are "red", "green", or "blue".

-s
: apply a sepia transformation

-m
: vertically mirror the first half of the image to the second half

-t:
: reduce the input image to a thumbnail based on the given scaling factor [1-8].

-n:

tile thumbnails of the input image based on the given scaling factor [1-8].

-o:
write output image to the specified file. Existent output files will be overwritten.

EXIT STATUS

`ppmcvt` exits 0 on success and 1 on failure.

EXAMPLES

`ppmcvt -o out.pbm in.ppm`
read in.ppm PPM file and write converted PBM file to out.pbm

`ppmcvt -g 16 -o out.pgm in.ppm`
convert the PPM image in.ppm to a PGM image in out.pgm

`ppmcvt -s -o out.ppm in.ppm`
apply a sepia transformation to the PPM image in in.ppm and output the new image to out.ppm

`ppmcvt -n 4 -o out.ppm in.ppm`
tile 4 1:4-scaled (quarter-sized) thumbnails of the image in in.ppm into a new PPPM image in out.ppm.

ERRORS

`ppmcvt` should print to the standard error output stream exactly the specified line and then exit under the following circumstances:

"Usage: ppmcvt [-bgirsmtno] [FILE]\n": malformed command line

"Error: Invalid channel specification: (%s); should be 'red', 'green' or 'blue'\n”

"Error: Invalid max grayscale pixel value: %s; must be less than 65,536\n"

"Error: Invalid scale factor: %d; must be 1-8\n"

"Error: No input file specified\n"

"Error: No output file specified\n"

"Error: Multiple transformations specified\n"

(File errors are handled for you by the provided `pbm` library.)

Implementation and Submission Details

You must implement `ppmcvt` according to the specifications given above. You are given skeleton files, `ppmcvt.c` and `pbm_aux.c`, in which to place your solution code. You may add helper functions to these files as you see fit.

We suggest you do your testing from within you "priv" directory. When you are ready to submit create a lab0 subdirectory of cs551 and copy just your `ppmcvt.c` and `pbm_aux.c` there.

The PBM Library (partially provided)

The given PBM library (`pbm.h` and `pbm.c`) does the following:

1. Defines structs for PBM, PGM and PPM image types;
2. Defines I/O routines to read/write the images from/to PBM, PGM and PPM files.
    a. (Note: The read routine does not handle image files with embedded comments.)
3. Declares memory allocation/deallocation routines for PBM, PGM and PPM structs.
    a. You must implement these routines in `pbm_aux.c`.

Image File Formats

PPM, PGM and PBM files are simple (and inefficient) ASCII text file image formats comprising a small header followed by integer values that represent each pixel in the image. Wikipedia has a good description here: https://en.wikipedia.org/wiki/Netpbm.

Image Transformations

Your program should produce *exactly the same output images* as mine. My program uses floating point arithmetic for all intermediate calculations then converts the resulting floats to integers as appropriate.

Bitmap:

To compute black and white bits from RGB pixels use:

$$Average(R+G+B) < PPMMax/2$$

Grayscale:

To compute grayscale pixels from RGB pixels use:

$$\frac{Average(R+G+B)}{PPMMax} \times PGMMax$$

Isolate:

For all pixels, set all but the specified "red", "green" or "blue" channel to 0.

Remove:

For all pixels, set the specified “red”, “green” or “blue” channel to 0.

Sepia:

For the sepia transformation, compute RGB pixels as follows:

$$NewR = 0.393(OldR) + 0.769(OldG) + 0.189\,x(OldB)$$
$$NewG = 0.349(OldR) + 0.686(OldG) + 0.168\,x(OldB)$$
$$NewB = 0.272(OldR) + 0.534(OldG) + 0.131\,x(OldB)$$

Mirror:

Vertically reflect the left half of the image onto the right half.

Thumbnail:

The height and width of the output thumbnail should be 1/n the height and width of the original image, respectively, where n is the input scale factor. Shrink the input image simply by outputting every n^{th} pixel in both dimensions starting with the first.

Nup:

Tile n 1/n scale thumbnails, where n is the input scale factor. The output image should be the same size of the input image.

Requirements and Constraints

This assignment aims to make you familiar with some ‘C’ programming basics. As such, we impose several requirements and constraints on your implementation:

1. You may not modify `pbm.h` nor `pbm.c`: you will not submit these files. We will compile your solutions using our original versions of these files.
2. You must use `getopt()` to process your program’s command line inputs.
3. You must use the provided pbm library (described below)
4. You may use only the following library or helper functions:
    a. C Memory Allocation: `malloc(), realloc(), calloc(), free()`
    b. Command line parsing: `getopt()`
    c. Other: `fprintf(), strtol(), strcmp(), exit()`
    d. PBM library
5. Intermediate storage: You must use dynamically allocated memory to store any intermediary image data. That is, you may not create temporary image files nor use static arrays (for example, `int image[MAXHEIGHT][MAXWIDTH]`). Instead, you should create an array like: `int **image` and dynamically allocate the precise memory needed depending on the image size.
6. You must free dynamically allocated memory immediately when no longer needed.