Practice 3: Threads

In this laboratory assignment, you will understand how to design and implement a multithreaded C program using POSIX threads (`pthreads`) to accelerate the histogram computation of a grayscale image stored in PGM (Portable Graymap) format. You are provided with a sequential implementation of the program, along with example images and a Python script for histogram visualization. Your task is to study the sequential version and develop a correct and efficient parallel implementation.

Provided Material and Usage

The files `P3_sequential.c` and `parsePGM.c` implement the sequential solution. The program reads a PGM image, computes its histogram, and writes the result to a text file. The histogram can then be visualized using the provided Python script `showHistogram.py`. You are also given a function `parse_pgm_header` which return the number of bytes of the header, as well as extract all needed information from it.

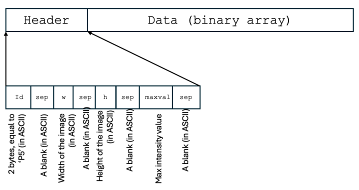
The sequential program can be compiled and executed as follows:

```
gcc P3_sequential.c parsePGM.c -o build/computeHistogramSequential
build/computeHistogramSequential Data/heart.pgm Data/histogram_heart.txt
python showHistogram.py Data/histogram_heart.txt
```

The `Data/` directory contains several example images. The file `heart.pgm` corresponds to an explanted zebrafish heart scanned at the ALBA synchrotron facility to study cardiac development during fetal stages. Both a full image and a cropped version (excluding high-intensity artefacts) are included.

The PGM Format

The PGM (Portable Graymap) format is a simple grayscale image format consisting of a short ASCII header followed by raw, uncompressed pixel data. In the binary variant (identified by the magic number `P5`), the header specifies the image width, height, and maximum gray value. These fields are separated by whitespace and may vary in length. After the header, the image pixels are stored sequentially in row-major order. If the maximum gray value is less than or equal to 255, each pixel is represented by one byte; otherwise, two bytes are used per pixel. In this laboratory, you may assume that all images use one byte per pixel (maximum gray value < 256).



To inspect the beginning of an image file, you may use:

```
head -c 500 Data/heart.pgm
```

Assignment Description

You must implement a multithreaded version of the histogram computation program using `pthreads`. The program will receive, as the last command-line argument, the number of worker threads to create. The main reads the image metadata, determines the size of the data segment (excluding the header), and creates the threads.

Each worker thread must process a distinct portion of the image data. The portion assigned to a thread is defined by a starting offset (measured from the beginning of the file, including the header) and a number of bytes to read. The main thread should indicate to each worker thread the portion of the image that it must process. Since the image size may not be perfectly divisible by the number of threads, the last portion may contain fewer bytes. The threads must compute the histogram, and once all threads have completed their execution, the main thread will write the complete histogram to the output file. You must avoid race conditions. For maximum credit, each thread must perform file reading using a buffer of at most 1024 bytes, as well as avoid unnecessarily locks.

Hint1: Because file descriptors maintain a shared file pointer, using a single shared descriptor across threads may lead to unintended interference. For this reason, it is recommended that each thread opens the file independently.

Hint2: The following structure may be used to pass arguments to each worker thread:

```
typedef struct {
    char* path;
    int offset;       // Offset from the beginning of the file (including header)
    int bytesToRead;
    // any other info you might want to pass
} ThreadInfo;
```

Questions

1) Discuss the race conditions of your solution, and where did you need to use locks to avoid them.
2) In your solution, how many threads can simultaneously read from the file? And update the histogram?
3) Report the needed execution time as a function of the number of threads. Comment your results.

Delivery instructions

1) Submit through gradescope before Sunday 22/02/26 (Group 102 will have delivery the 01/03/26, since the practice class is later). You will be given output on whether your code compiles or not in the gradescope system.
2) Add all the integrants of the group during the submission. Here you have the instructions.
3) The submission requires to **exactly** follow the structure below:
    a. *src* folder, with all the sources. There can only be one C code with a main.
    b. compile.sh, a bash script which will contain the compile command (and no other commands).
    c. integrants.txt : the name and NIA of all the group members
    d. P3_report.pdf: report answering the questions
4) Remember to remove all hidden files before submitting.
5) Non adherence to the submission instructions will result in point penalization.