PRACTICE 3 OPERATING SYSTEMS QUESTIONS

Gina Domech
Maria Aran

1. **Discuss the race conditions of your solution, and where did you need to use locks to avoid them.**
   A race condition occurs when multiple threads access shared data concurrently and at least one of them modifies it, which results in unpredictable results. In our program, the shared resource is `globalhist[HISTSIZE]`, which is the global histogram array. Two or more threads could try to update the global histogram at the same time. We handled this using mutex. So, each thread computes the local histogram independently, then, the mutex is used to lock the section where threads merge their local histogram into the global histogram.
2. **In your solution, how many threads can simultaneously read from the file? And update the histogram?**
   Each thread opens the file independently and reads from its assigned offset, thus, all threads can read at the same time, but only one at a time can update the global histogram
3. **Report the needed execution time as a function of the number of threads. Comment your results.**

```
maria@DESKTOP-V1B8M2M:/mnt/c/Users/maria/Desktop/UPF-OS2026-MARIA-GINA/P3_MARIA_ARAN_GINA_DOMECH$ ./computeHistogramParallel Data/heart.pgm output1.txt
./computeHistogramParallel Data/heart.pgm output2.txt 2
./computeHistogramParallel Data/heart.pgm output4.txt 4
./computeHistogramParallel Data/heart.pgm output8.txt 8
Execution time: 1.932186 seconds
Execution time: 1.264563 seconds
Execution time: 1.025553 seconds
Execution time: 0.771812 seconds
```

1 thread	1.93 seconds
2 threads	1.264 seconds
4 threads	1.0255 seconds
8 threads	0.7718 seconds

In conclusion, using multiple threads reduces the execution time because the work of reading pixels and counting their values is split among several threads that can run simultaneously. Each thread processes its portion of the image independently, using its own local histogram.