Pre CQI 2023 - Operating System - 10 pts

Context

The Operating System challenge serves to evaluate your competencies related to Operating systems, mon particularly, parallelism. You will have to accelerate the execution of a video editing program with the help of parallelism.

Software Required

- Linux
- cmake
- pthreads
- tbb-dev
- gcc and g++
- libPng

Delivery

1. Use the `make remise` command
2. Submit the `remise.zip` file

Code provided

- `source/main.c`
  - Contains the program entry point that processes command line arguments and starts the desired image processing pipeline.
- `source/image.c` `include/image.h`
  - Contain structures and code for reading/writing PNG format images.
- `source/filter.c` `include/filter.h`
  - Contains different functions allowing filters (modifications) to images.
- `source/queue.c` `include/queue.h`
  - Contains a simple implementation of a queue allowing reading/writing by several execution nodes. These structures and functions are strongly recommended when implementing the pipeline using pthreads.
- `source/pipeline-serial.c`
  - Contains a serial reference implementation of the pipeline.
- `source/pipeline-pthread.c` **(TO BE COMPLETED) (5 POINTS)**
  - Contains the requested parallel implementation of the pipeline using pthread
- `source/pipeline-tbb.cpp` **(TO BE COMPLETED) (5 POINTS)**
  - Contains the requested parallel implementation of the pipeline using TBB
- `data/fetch.sh`
  - Contains a script to download test images.
- `data/check.sh`
  - Contains a script to check if the produced images are identical to the serial images.

Specifications

The pipeline to be implemented must contain the following stages: 1. Read an image 2. Double the image size 3. Convert color space from RGB to HSV 4. Gradually shift the H component by 4 with each successive frame 5. Convert color space from HSV to RGB 6. Save Image

Compile

To compile the application, it is recommended to create a `build` folder at the root of the project in order to separate the generated files.

```
$ mkdir build && cd build
```

Then, we configure the project with `cmake`. This one can therefore be compiled with `make`

```
$ cmake ..
$make
```

It is not necessary to re-execute all the commands above to recompile the binary, only the last one. You can run `./pipeline --help` to see program options.

Orders

The `Makefile` generated by `cmake` contains the special commands below.

- `make format`
  - Use `clang-format` to format source code.
- `make discount`
  - Creates a **ZIP** archive containing the files for delivery.
- `make run-serial`
  - Run the `serial` pipeline with the data in the `data/` folder and measuring the elapsed time.
- `make run-pthread`
  - Run the pipeline using `pthreads` with the data in the `data/` folder and measuring the elapsed time.
- `make run-tbb`
  - Run the pipeline using `TBB` with the data in the `data/` folder and measuring the elapsed time.
- `make run-all`
  - Runs the above 3 pipelines

Test

```
$ ./data/fetch.sh // Download images
$ mkdir build && cd build && cmake .. // Generate the makefile in the build folder
$ make run-all // Compile and run the 3 algorithms
$ ../data/check.sh // Check if all images created are identical
```