Lab

Synchronization I

For this lab you are given five programs that demonstrate the use of various `pthread` commands and synchronization operations. Work through the underlined exercises.

- `pgm1.c`: The main thread creates two new threads. The main thread writes whole numbers in a file called *whole_num*, one thread executes the `odd` function which writes odd numbers in a file called *odd_num*, and the other thread executes `even` function which writes even numbers in a file called *even_num*.
- `pgm2.c`: Same as `pgm1.c` except that the threads are created with specific parameters instead of default parameters, and write to `stdout`. This implementation demonstrates the fact that `pthreads` can have different *attributes*, e.g., a `pthread` can be either *joinable* or *detached*. You do not need to be aware of all the `pthread` attributes to complete the programming assignment, but to fully understand this code, it will require a little research into `pthread` attributes on your part.
- `pgm3.c`: Same as `pgm2.c` except that the threads use mutexes to synchronize their writes. Explain the difference in output between `pgm2.c` and `pgm3.c`.
- `pgm4.c`: A solution for producer consumer problem using `pthreads`. In this program the buffer is of size 1. Modify `pgm4.c` for a buffer of size larger than 1. One you have done this, modify `pgm4.c` for a variable number of producers and consumers.
- `pgm5.c`: An *incomplete* solution for dining philosophers problem using `pthreads`. Complete this to a solution for the dining philosophers problem by filling in the comments with the correct wait/signal semaphore operations. See the semaphore handout for more details.

Run `make` to compile, or use the following command for each program:

```
gcc -o <object_filename> <filename> -lpthread
```

As always, you are encouraged to work with others and discuss this lab with your peers on Piazza, especially if you have any issues.