1 WriteUp 1

1.1 Concurrency

1. The objective of concurrency is to understand how to syncrenize operations between two tasks.
2. We use pthreads for multithreading and mutexes for critical sections. One buffer, two threads (plus the main thread), and one mutex one mutex played a role in producing and consuming jobs. Details regarding the role of each are described below:

   Buffer The buffer is circular FIFO queue, containing jobs awaiting to be consumed. The buffer's starting and ending points are marked by `jhead` and `jtail` variables. Produced jobs are appended to the circular queue, incrementing and wrapping `jtail`. Consumed jobs are removed out of the cicrular queue, incrementing and wrapping `jhead`.

   The buffer has a limit defined by `NUM_JOBS` preprocessor. Once `jtail` reaches the limit, that is one less than `jhead`, the producer stops creating jobs until space becomes available again.

   Producer The producer thread is a continuous while loop, which does the following:

   (a) Acquire mutex via `pthread_mutex_lock`.
   (b) Check if the jobs queue is not full. If the queue is full, simply unlock the mutex and continue the loop.
   (c) Create a job with randomized wait time and number values. The randomization is achieved by `rdrand` if supported by platform or `mt19937` if not.
   (d) Append job to the queue, incrementing and wrapping `jtail`.
   (e) Release the mutex via `pthread_mutex_unlock`.

   Consumer The consumer thread is also a continuous while loop, which does the following:

   (a) Acquire mutex.
   (b) Check if queue is not empty. If empty, release the mutex and continue the loop.
   (c) Get job at `jhead`.
   (d) Print out the number value of job.
   (e) Copy job wait time to a seperate variable, `dt`.
   (f) Increment and wrap `jhead`.
   (g) Release the mutex.
   (h) Sleep for await time indicated at `dt`.

3. Hi
4. Although I had previous experiance with mutexes and multithreading, I did learn that concurrency can as well be achived with semaphores.

1.2 Version Control

1.3 Work Log