Name: Ethan Mok
ID: #811383355

Design of the multi-threaded server:

In my program, the first thing that I did was create the number of threads as specified by the command line argument. I used an array to act as my thread pool, storing all of my threads inside for ease of access. I then used a for loop over that array to create a thread and stored it into the pool. This thread pool acted as my consumers. Each consumer thread called a function I made called req_handler(). This function removes an item from the buffer using a critical section and consumes that item. The other thread in the program is the listener thread. This thread is outside my consumer thread pool. The thread calls the req_handler method that I slightly modified to place items into the shared buffer using a critical section.

The two aforementioned critical sections use a semaphore and a mutex lock in order to avoid race conditions. Following the consumer-producer model, I used sem(empty) and the lock to lock the producer critical section, which placed an item in the buffer, and I used sem(full) and unlock to release it. I then used sem(full) and lock to lock the consumer critical section, which removed the item inside the buffer, and I used sem(empty) and unlock to unlock the critical section.

By locking the section in this manner, the listener process will fill the buffer unless it is already full and the consumer process will empty the buffer as items are placed in. There will also be no race conditions because I locked the shared out variable in the req_process() function. This means that threads will not accidentally pull the same item out of the buffer.

Design of the crash handler:

My main() function uses a while and for loop to continually check to see if a thread has exited. The exit status of a thread, which is the return value of the pthread_tryjoin_np() function, is placed in a variable called error. If that error value is equal to zero then the thread “crashed” or has exited and a new one is made and added back into the thread pool. This is how the crash handler deals with exited threads.