READ ME

I synchronized my code by locking down global variables and shared data structures.

My Buffer.cc class makes use of mutex and conditional variables. When a client is appended it locks and signals that the queue is not empty.

```
pthread_mutex_lock(&lock );
bufferQueue.push(client);
pthread_cond_signal(&not_empty);
pthread_mutex_unlock(&lock);
```

When a client is taken out of the queue, a while loop checks if the queue is empty or no client is available, then a conditional variable signals not empty and locks, causing the thread to sleep until a client is ready. The queue then pops the client from the front and unlocks.

```
pthread_mutex_lock(&lock );
while(bufferQueue.empty()){
        pthread_cond_wait(&not_empty, &lock);
 }
 int client = bufferQueue.front();
 bufferQueue.pop();
 pthread_mutex_unlock(&lock );
```

My server.cc class makes use of a mutex lock and unloack function in the serve function. This only allows one thread at a time into the handle function to handle a single request at a time.

```
buffer.append(client);
pthread_mutex_lock(&lock);
handle();
pthread_mutex_unlock(&lock);
```