The Problem Spec:
Create 2 program types which implement the client server model.
Program 1: Server

1. Creates a server to accept client requests for reading/writing privileges
2. ensures only one program has privilege when writing.
3. ensure only readers have privilege when reading

Program 2: Reader/Writer

1. Creates a reader or writer client which can request privileges from the server
2. request privilege msg
3. On privilege write/read from/to the requested file.
4. release privilege msg
5. repeat

The difficulty with this program is setting up a unix server and ensuring a fair mutual exclusion

Brief Algorithm:

1. Server accepts clients as they connect and an empty list of filegroups is created
2. On requesting msg the msg is added to a queue of requests for its group if the group does not exist the group will be created.
3. After any msg the request queue is checked and if there are no ME conflicts ie(already a writer or multiple readers on write request) a ticket is sent to the client to allow reading or writing and increment the current number or readers/writer
4. On release message the current number of readers/writer is decremented to allow other clients to get ticket

This is a decent solution because it will avoid race conditions, allow multiple readers to read at the same time and not starve the writer threads when they want to write through the use of the request queue.

How to Run:

1. cd to this folder
2. run make
3. run ./server
4. open new terminal
   cd to this folder
   run ./rw {UNIQUE CLIENT ID} R {GROUP FILE} {SERVER MACHINE IP}
5. open new terminal
   cd to this folder
   run ./rw {UNIQUE CLIENT ID} W {GROUP FILE} {SERVER MACHINE IP}
6. repeat step 4 and 5 to create as many readers/writers as desired

Tests:

1. 5 groups of clients who access files A, B, C, D and E respectively. Each group consists of 3-6 client processes with 30 percent of writers
2. 1 server 1 group 1 writer
3. 1 server 1 group 5 writers and readers
4. 1 server 5 group 5 writers and readers per group
5. 1 server 5 group 5 writers

All tests ran successfully with no client starvation due to the request queue