Client Side

On the client side I had it read in the server options from a file and then try to connect to those servers. If it got a response then that server was running otherwise I skipped over that server. Once all the servers are known I started a thread for the number of chunks passed in from the command line. Each thread is given a socket that corresponded to one of the servers. Once the threads are running they send the offset of the file and the amount that it wants to receive. After that it starts receiving the data from the server and sends an ack back. Once it has gotten all the data it writes it to the file. Once all the threads are dead the program exits.

Server Side

On the server side the server would sit in an infinite loop and wait for a command to be received. If the command was for a file it would first find the size of the file and then send back its size, if the file didn't exits it would send back an error. If the server received the offset command it would start up a new thread to listen to the socket and send back to information to the client. The thread allows for the main thread to still listen to new clients. When the child thread send the data to the client it waits for a response otherwise it will try to send the data again. After the segment of the file has been sent the thread exits and the main thread continues listening for new commands.

Running my code.

The client tries to delete the file passed into it just so that there is no confusion at start up. After that it will write the file to the directory it is running in. When using the loop back address the server and client must be run in different directories. To use my test, the test will go into the root directory along with the file that wants to be transferred. The file name must be specified in the script. The servers will be started in the root directory while the client will be in the src directory. When the program finished the new file will be in the src directory and the test file will call diff on it.