How to run:

```
./server <IP> <big> <med> <small> <offsets>
./display <IP>
./control <IP>
```

Known Bugs:

I used seconds instead of lines per minute for timing because it seemed simpler. For lines per minute functionality you can convert it into seconds :)

There is zero functionality to safely quit and reconnect

Commands:

pause: pauses the sending to display

resume: resumes the sending to display

pos <int 0-maxPos>: goes to the chunk starting at 0. Will change after chunk is finished

version<0-3>: 0 is highest “quality” then descending order.

The display and control are just built upon Beej’s server-client examples. They simply loop forever either waiting for user input to send or to recv and print from the server.

The server First sets up ports for the display and control and waits for them to connect. Once they connect we loop forever sending and receiving. We first seek to the position on the version that is in memory. Initially this is zero and the highest quality if the position is less then the maximum position, if it is we simply wait for commands so hopefully the control moves the pos to a better place.

Then we are in the Sender function which get a line if we are not paused, check if it is the end of a chuck if so we update the position and go to handle.

Handle is basically a select statement with a timer. Until the timer runs out we wait for a command from control and after that we’ll send a line to display.

recvControl receives commands from control and tokenizes them and then sends them to doCommand which is a bunch of if statements and updates global variables.

Testing:

I used GDB, and manual testing. I used the commands in different combinations and tried different files to see what would happen