Name: Stephen Paradis
Date: May 13 2016
Current Module: Operating Systems
Project Name: Relay

Project Goals:

The goal of this project was to create a dispatcher(server) that could accept multiple connections from listeners(clients). The dispatcher also needs to end when it receives the EOT character which for my project is “q”. The listener is supposed to echo whatever the dispatcher sends it, but not any keyboard interrupts.

Considerations:

- The dispatcher should be able to have multiple connections from listeners
- The dispatcher should be able to send messages to multiple listeners
- The dispatcher should quit when given EOT character
- The listener should print whatever it gets from dispatcher
- The listener should echo any keyboard output to the screen other than what dispatcher sends
- Both programs should be made when make all is called

Initial Design:

I decided to use sockets and a client server model to do this project. I made my dispatcher a server type file and my listeners are my clients. For the dispatcher my EOT is ‘q’ which quits the server and sends 0 to the listener. On the listener when it receives the 0 it tells the user that the connection has been refused. For the problem with making it useable in different directories and by different users I put my SOCK_PATH file in my directory chmodded with 0777.

Data Flow:

The flow starts with the dispatcher running and setting up its socket. It then binds and listens to connections from listeners. The listener runs connect and once it's connected it listens for anything the dispatcher sends out. If the dispatcher sends something out the listener will print it to the screen unless it is a keyboard interrupt.

Communication Protocol:

- None

Potential Pitfalls:

- If the program is using sockets, it could not be chmodded at the top level for multiple people to run it.
- Not handling sending messages to all listeners.

- Not connecting correctly to all listeners

Test Plan:

User Test:

1. Ran listener without dispatcher
2. Rand dispatcher by itself
3. Wrote text into listener
4. Ran dispatcher and one listener
5. Ran dispatcher and multiple listeners
6. Sent text from dispatcher when one or more listeners were attacher
7. Set eot (‘q’) to dispatcher
8. Tested what happened when server quit

Automated Test:

- None

Test Cases:

1. Ran the listener without the dispatcher running to see if it would gracefully tell the user that the connection has been denied.
2. Ran the dispatcher by itself to see if it would run and wait for connections.
3. With both dispatcher and listener running wrote text into listener to see if anything happened to dispatcher.
4. Ran dispatcher and one listener to see if they connect correctly.
5. Checked if the dispatcher could send messages to every listener when more than one is connected.
6. Checked to see if the dispatcher could write and send messages to only one listener.
7. Typed q as my eot character to see if dispatcher quits gracefully.
8. Tested to see if listeners quit successfully when server quits first.

Expected Result:

I expect the dispatcher to create a socket and to listen for 5 connections and to send messages to whoever is connected. I expect the listener to connect and echo any messages received from the dispatcher to the screen.

Conclusion:

At the beginning I started using sockets which in the end seemed to be the right path however the initial way i was specifying port and address did not work. In the end I had to changed away from that method and use sets and select() to create a list of connection. This allowed me to send messages to each connection or add connections in the middle before sending the message. In the future it might be good to change how the socket path file is saved to allow for use in different directories and by different users.