Name: Daniel E. Ems
Date: 05/13/2017
Current Module: Operating Systems
Project Name: Relay

Project Goals

The goal of the project was to establish interprocess communication between two programs, dispatcher and listener. The dispatcher hosted clients and transmitted data. The client connected and received data being transmitted.

Considerations:

- What approach best serves this project.
- What form of sockets should best if used.
- What threads might be needed inside programs.
- What attributes does the listener need to possess.
  - Receive transmission.
  - Connect.
- What attributes does the dispatcher need to possess.
  - Accept connections.
  - Send transmission.
  - Handle user input.
  - Properly exit and shut down transmission.

Initial Design

The initial design of the project utilized AF_INET sockets. The dispatcher served as a server, and the listener a client. The communication between programs utilized the sockets to transmit user input over the loop back address (127.0.0.1) on port 51236. The server allowed for multiple connections and only exited on EOT or a sigint signal.

Data Flow:

The dispatcher serves as a server for socket connections. The server creates, binds, listens and accepts socket connections. The server will stay in an accept block until a connection is established. The file descriptor that is returned is passed to a thread that handles user input, and then re-enters an accept block. The thread that handles user input, will wait for the user to input data over stdin. The thread will then iterate through an array of connected file descriptors and send the user data.

The listener sends connection requests. The listener will create a socket struct and and wait to receive user input. Once the listener receives a transmission, the program prints the transmission to the listener screen. The listener then renters a waiting state for future transmissions.

Potential Pitfalls

The largest pitfall about this project is dealing with certain unknown/uncontrollable behaviors of AF_INET sockets. For example TIME_WAIT is set on most sockets to handles packets that are possibly still being sent over the network. This creates a slight delay between runs of the programs as you wait for TIME_WAIT to expire. While this is a merely a convenience issue. It does highlight a

greater issue of external factors and protocols set on some of the tools utilized that are out of our control. Unless we were administrators.

In addition to unknown/uncontrolled behavior, sockets are difficult and very complex. Understanding how to ensure they are properly set up, and closed, is crucial in minimizing the number of potential errors that could arise. Even when done so properly, there are issues that are unique to sockets and must be dealt with in different ways. For example, exiting an accept block to properly exit your program.

Finally, once connections are successfully established, there is still the issue of addressing good development practices. This requires the use of additional tools and skills such as threads. While these are also new concepts, the real challenge is being sure that their implementation does not adversely affect the potentially delicate behaviors described above. However, like most things in development; testing will reveal all.

Test Plan:

User Test:

1. Try and create a socket.
2. Compile the program with no errors in regards to the socket(socket(),bind()).
3. Try and establish an ip and port to use, declared in the struct.
4. Try and create a socket that listens implementing (listen(), accept()).
5. Try create a socket that connects utilizing steps 1 and 2.
6. Try establishing a connection using errno flags.
7. Try transmitting data using send() and recv().
8. Try looping fgets and recv so that data can be sent continuously.
9. Try threading fgets.
10. Try looping accept to allow for multiple connections.
11. Try opening multiple terminals for multiple transmissions.
12. Try shutting down the program appropriately.

Test Cases:

NOTE: All test cases will be in reference to the transmission of data after the socket connection has been established. This is done to avoid redundancy. For many test cases for setting up the sockets dealt with eliminating warnings and compiler errors. The functionality of the sockets and the program can not be tested until connections are met.

1. Hard code “hello” in a single transmission to the client.
2. Print receiving transmission to the screen in a single client.
3. Accept user input via fgets, and send to the client.
4. Repeat step 2.
5. Try and loop accepting user input and sending to a single client.
6. Continue to repeat transmissions and not spam client terminal.
7. Allow two clients, and send a message.
8. Address issue of not all clients receiving the same transmission.
9. Place file descriptors in an array and try iterating through them as you send.
10. Try sending to multiple connections.
11. Try to ctrl-c out of the program.
12. Implement sigaction to handle signals.

13. Try to send EOT to the program.
14. Try to properly close thread when EOT is received.
15. Try to properly exit accept block when EOT is received.
16. Try and close the program with no valgrind errors with EOT.

Conclusion

The project was a lot of fun. Sockets are interesting, challenging to learn and fun to implement. Being able to implement AF_INET sockets was fun in establishing connections on separate boxes on the same network. This made interprocess communication fun. There are many nuances to sockets that you must discover and address along the way. However, there is a lot of support and material to help you get through any issues you might have.

After sockets were established, and attempt at mmap was attempted, and ultimately abandoned. The idea was to have the dispatcher create a shared memory object, map it to it’s address space and then write to it. Then, clients would map the same object to their space and read. The issue that ultimately led to the abandonment was futexs. There is little support on how to properly implement futexs and this was an impassible issue. This led to the final implementation of AF_INET sockets. A tool that will serve valuable for years to come.