Operating Systems Lab 4 report

Joshua Marple

September 2014

Introduction

1

My program controls when the client and server can run by requiring (first) the proper setup- server waits for client connection, client searches for server, then connects, etc.

When transmitting the actual messages, they play a bit of hot potato. First, the client sends something to the server then waits for a message. The server takes the message from the client, capitalizes it, prints it out, then sends it right back. This continues 3 times.

2

The handshake socket is used to accept connections. Rather than blocking execution while waiting on a direct connection, clients connect to the handshake socket instead, which then facilitates direct conncection between the two sockets. The handshake socket also differs from the data transfer socket in that it doesn't listen for data transmission. Instead, it listens for incoming connections.

3

Sockets possess a number of advantages when compared to pipes. First and most importantly, by using sockets instead of pipes, our program would be as easily adaptable to work over a network as it is between two processes. Nowadays, this sort of communcation is just as common and likely a use case as would be interprocess communcation, and should be treated as importantly. A pipe would be wholly unsuited for use in a network. The second would be that sockets are bidirectional, whereas pipes are not. To create this same program using pipes, I would need to open up a second pipe instance.