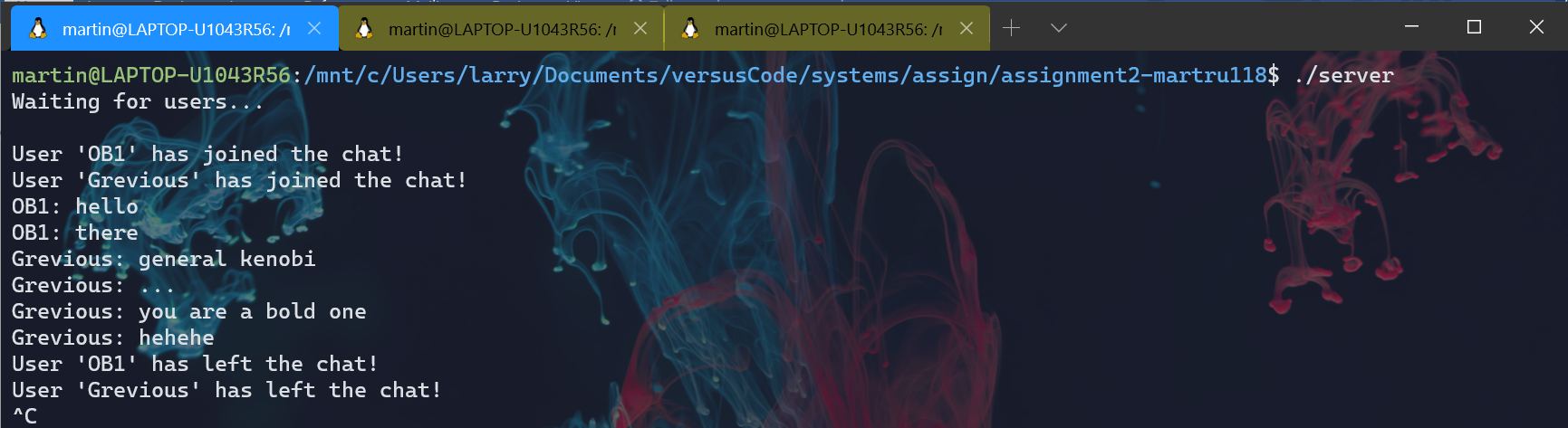
Assignment 2: Chat Client & Server

# Server

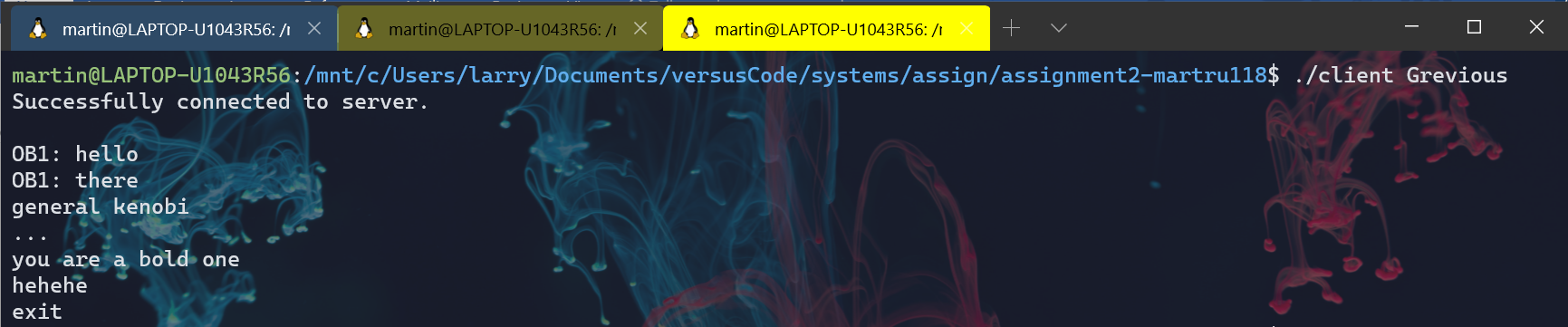


First, establish a connection to the network via port 55555. To do this, we retrieve the file descriptor for the socket at this port by using the first address found by getaddrinfo. Here, we want a connection stream, so we want to find an address that would accept all connections on a host. Afterwards, we use setsockopt to make this address reusable after the socket has closed. Once we have a socket, we use bind to associate that socket with the chosen port.

Now, the server waits for incoming connections. Using listen, the server would be expecting up to 10 connections on the socket. These connections are then monitored using select. To read incoming messages, add the socket to the set \*readfds, a parameter of select. On success, the socket would be used for reading data.

Incoming connections are handled by the connAccept procedure. Here, the server retrieves the name of the client. The socket of the new client is added to the set allfds by keeping track of the maximum file descriptor. If there are no incoming connections, read data from a client. We use readn to receive messages. Then, we format the message to include the name of the client and send the formatted message to all other clients using writen.

# Client



The client connects to the network in a way similar to that of the server. The client retrieves a connection stream, finds an address that would accept all connections on a host, and uses that address to find a socket. The client also uses select in a similar way.

However, a connection is established using connect, and the port number is specified with htons. Before connecting to the network, we must check for command line arguments to make sure the client has a name. The client sends their name to the server after a connection is established. We now check all other clients for any messages. Then, we send a message to the server from keyboard input using stdin.