Ex8 - Networking

COP4600

Slides by: Ernesto A. Mujica

Socket Programming in C++

- Socket programming usually takes the form of a server-client model, where a server listens for a connections and a client connects to the server.
- In C++, socket programming requires a lot of setup. Refer to this link for how to set up C++ sockets for your server.
- In this exercise, you are only responsible for programming the **server**, the **client** program already exists (and is already installed on your VM).
- You will be using the **netcat** command to connect to your server for testing, using the following command:

  netcat localhost <port-num>

Socket Programming in C++ (cont.)

(Refer to the link in previous page on how to setup the functions below.)

- socket() will create the file descriptor for the server socket.
- setsockopt() will set the socket options for the provided socket.
- bind() will bind a socket to a given port and address structure.
- listen() will set the program to listen for an incoming client connection.
- accept() will accept the incoming connection and return a file descriptor used to communicate with the client socket.
- close() can be called on client socket to close the current connection. You can call listen again on the server socket to accept another client connection.
- close() can also be called on the server socket to disallow all future connections.

Socket Programming in C++ (cont.)

- Sending text to the client socket will be done using the send() function. The send function has the following structure:

  send(file_descriptor, ptr_to_data, amount_of_bytes, flags);

- For example:
  char message[ ] = “Hello, I am the server!”;
  send(client_fd, message, strlen(message), 0);

- Will send the given message to the client, where client_fd is the file descriptor returned from the accept() function. You can leave the flag parameter as 0 permanently.

- Receiving text from the client to the server will be done using the read() function, similar to how you used it in Ex7 to read from pipes.
  char message[10];
  read(client_fd, message, 10);

- This will take in 10 bytes from the client, and store them at pointer ‘message’.

Socket Programming in C++ (cont.)

- Before moving on to the rest of the exercise, it is recommended to start very simple.
- Begin by making sure you can establish a very simple connection between **netcat** and your **wallserver.** See if you can send a single simple message back and forth between the server and client, before you start adding more complexity.
- **Use the provided message header.** This exercise has a lot of strings that are case sensitive and we will be testing you based on your standard output, meaning that if things don't match exactly you will not receive a good functionality grade. Use the provided strings where appropriate.
- Message length for a given message will include the name of the messenger as well as the colon and whitespace after the messenger's name.
- For example, if the max message length is 80, and the messenger is "John", the prefix "John: " will take up 6 characters, meaning the message can only be a max of 74 characters.

Socket Programming in C++ (cont.)

- (Go over PDF, Messages.h, and demonstrate functionality.)