Eng 67: Lab 3

Lab 3 Learning Objectives:

1. To understand the basics of Internet programming and addressing.
2. How to create a program using UDP, an Internet *datagram* protocol.
3. Expand knowledge of C and POSIX programming with timers and string pattern matching.

TASKS

ENGS67: Implement a simple UDP client and server that function as follows:

- The client accepts a ***server*** IP address and port number as command line arguments. It repeatedly takes integers from the keyboard, sends them to the server, and prints the response from the server.

- The server takes a single command line argument specifying the port to receive messages on. It maintains a counter that is initialized to zero (0) and accepts messages from the client. If the content of the message is zero (0), then the counter is *reset* to zero. If the content of the message is a number, then it is added to the counter. The server responds to every message received with the current value of the counter.

You should initially test these programs by running them concurrently on the same host, for example:

```
% ./server 8888 &
% ./client 127.0.0.1 8888
```

Note that 127.0.0.1 is the loopback IP address associated with the current host. The server is run in the background and the client in foreground. **Subsequently, ensure that you are able to run multiple clients and the server on different hosts, e.g. hub1,hub2,hub3, etc.**

ENGS 115: Implement a simple chat client and server that provides a single chat room. The chat client should take an identifier (ID), server IP and port number as command line arguments. It should repeatedly take strings from the keyboard and support the following commands (signified by / as the first character):

/ping – queries the server to determine if it is up and prints the result
/join – joins the chat room
/leave – leaves the chat room
/who – obtains the current list of ID's in the chat room.

The server should:

- Respond to the ping messages to indicate that the server is up.
- Discard all messages from users who are not in the chat room.
- Broadcast chat messages to all members of the chat room except the originator.
- Prevent two chat clients from using the same ID.

Suggestion: *Initially implement ONLY the ping message. Subsequently, add other messages one at a time.*

NOTE: YOU ALREADY HAVE A RE-USABLE LIST MODULE FROM LAB 1**.**

You should initially test these programs by running them concurrently on the same host, for example:

```
% ./server 8888 &
% ./client steve 127.0.0.1 8888 &
```

Note that 127.0.0.1 is the loopback IP address associated with the current host. **Subsequently, ensure that you are able to run multiple clients and a single server on different hosts, e.g. hub1,hub2,hub3, etc.**

TOPICS

Research the following concepts:

- Watch the LAB 4 video
- How do you designate TCP rather than UDP protocol to use when you create a socket.
- What do the following functions do and where are they used (client or server):
    - connect(…)
    - listen(…)
    - accept(…)
    - send(…)
    - recv(…)
    - alarm(…)

- The accept(…) call blocks waiting for what condition to occur?
- Unlike UDP, TCP is a byte-stream protocol: send boundaries are not preserved. What does the *receiver* need to do to cope with this issue?
- How does a server know that a client has terminated and closed a connection?
- How might you interrupt a blocking recv(…) if there has been no message for 3 seconds?