LINUX CLIENT-SERVER DOCUMENTATION

COMP 4981 – LINUX Sockets

Spenser Lee A00925785

Thomas yu a00915638

Contents

[User Manual 2](#_Toc446552389)

[Application Overview 2](#_Toc446552390)

[Application Setup 2](#_Toc446552391)

[Application Usage 2](#_Toc446552392)

[Bash Script For Testing 3](#_Toc446552393)

[State Chart Diagram (Server) 4](#_Toc446552394)

[State Chart Diagram (Client) 5](#_Toc446552395)

[Pseudo Code 6](#_Toc446552396)

[Client 6](#_Toc446552397)

[Pseudo Code (continued) 7](#_Toc446552398)

[Server 7](#_Toc446552399)

[Testing 9](#_Toc446552400)

# User Manual

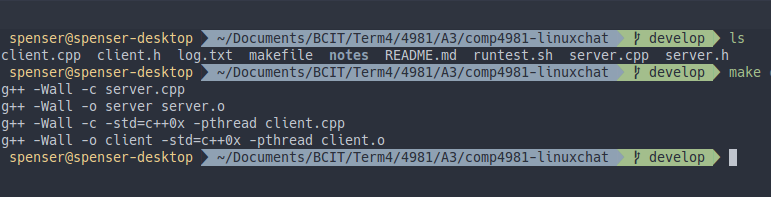
## Application Overview

This application demonstrates the use of sockets and the select call with a server and client chat program. Multiple client connect to a server and send messages; each message sent to the server is then echoed to all other clients.

## Application Setup

To compile the client and server executables:

1. Open a terminal and navigate to the folder containing the source files
2. Execute the command “make”

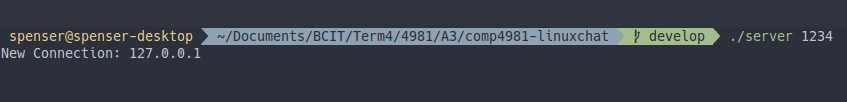


## Application Usage

With the applications compiled you can run multiple clients and a server.

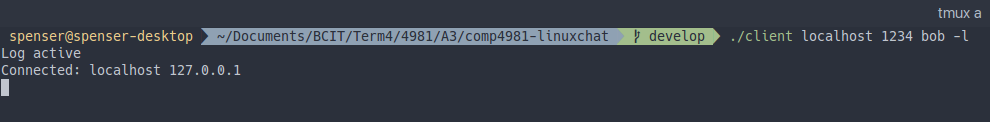
First launch a server, in a new terminal window execute the following command: **./server [port]**

You can optionally choose a port to run on (default 7000).

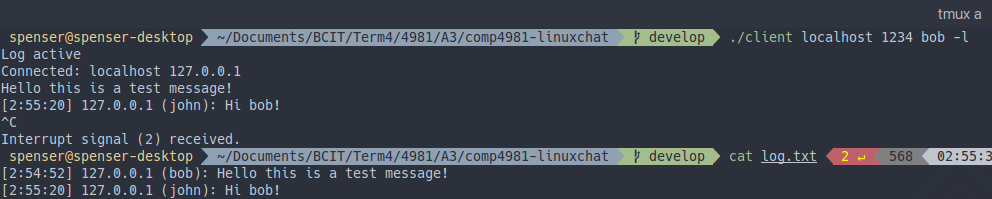


To launch a client, execute the following command: **./client hostname [port] [username] [-l]**

You must enter a hostname, and can optionally choose a port (default 7000), set a username, and enable logging with -l.



With the client running, enter a message and press enter.



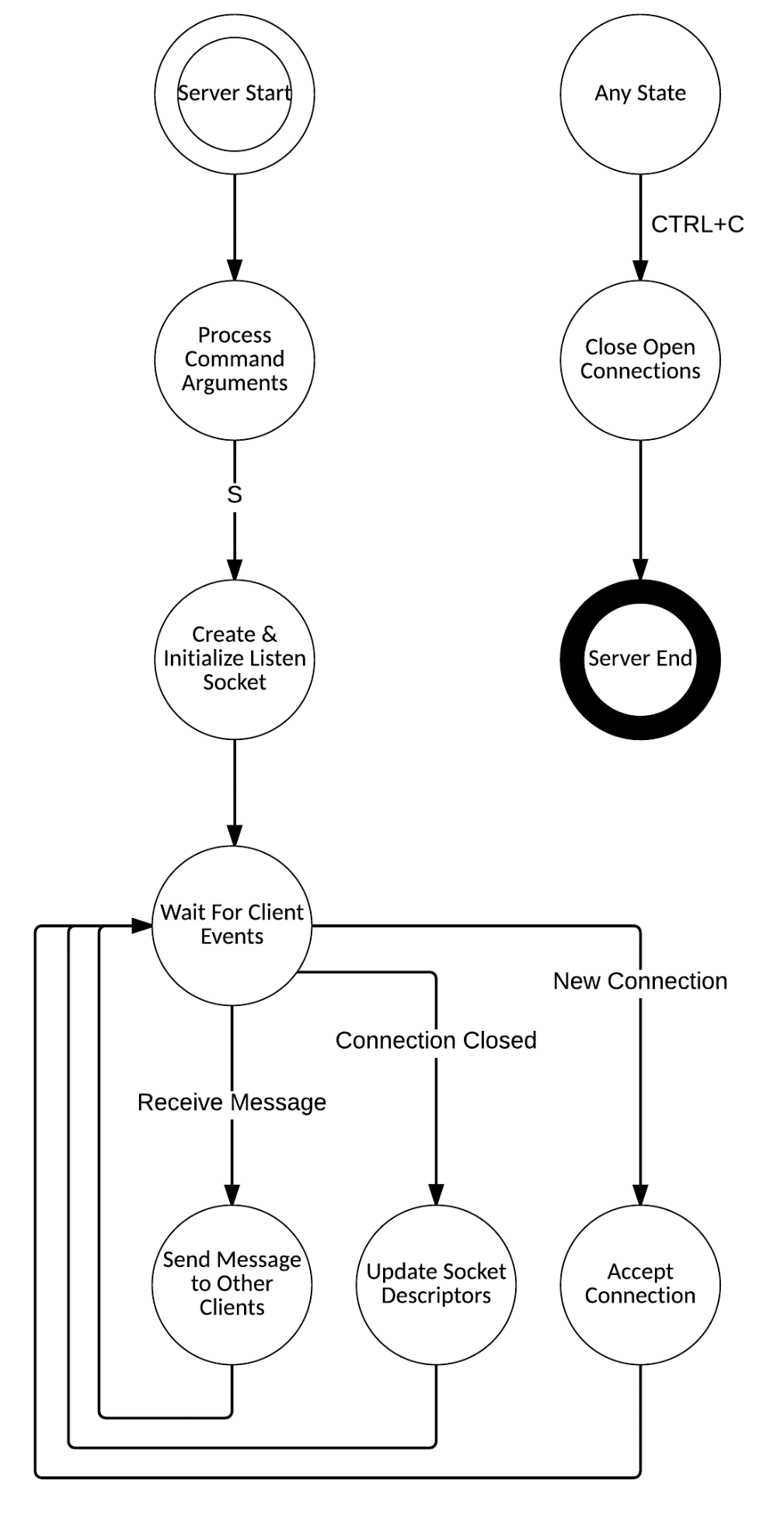
## Bash Script For Testing

To aid with testing, I also created a script which will launch a server and 3 named clients applications on localhost.

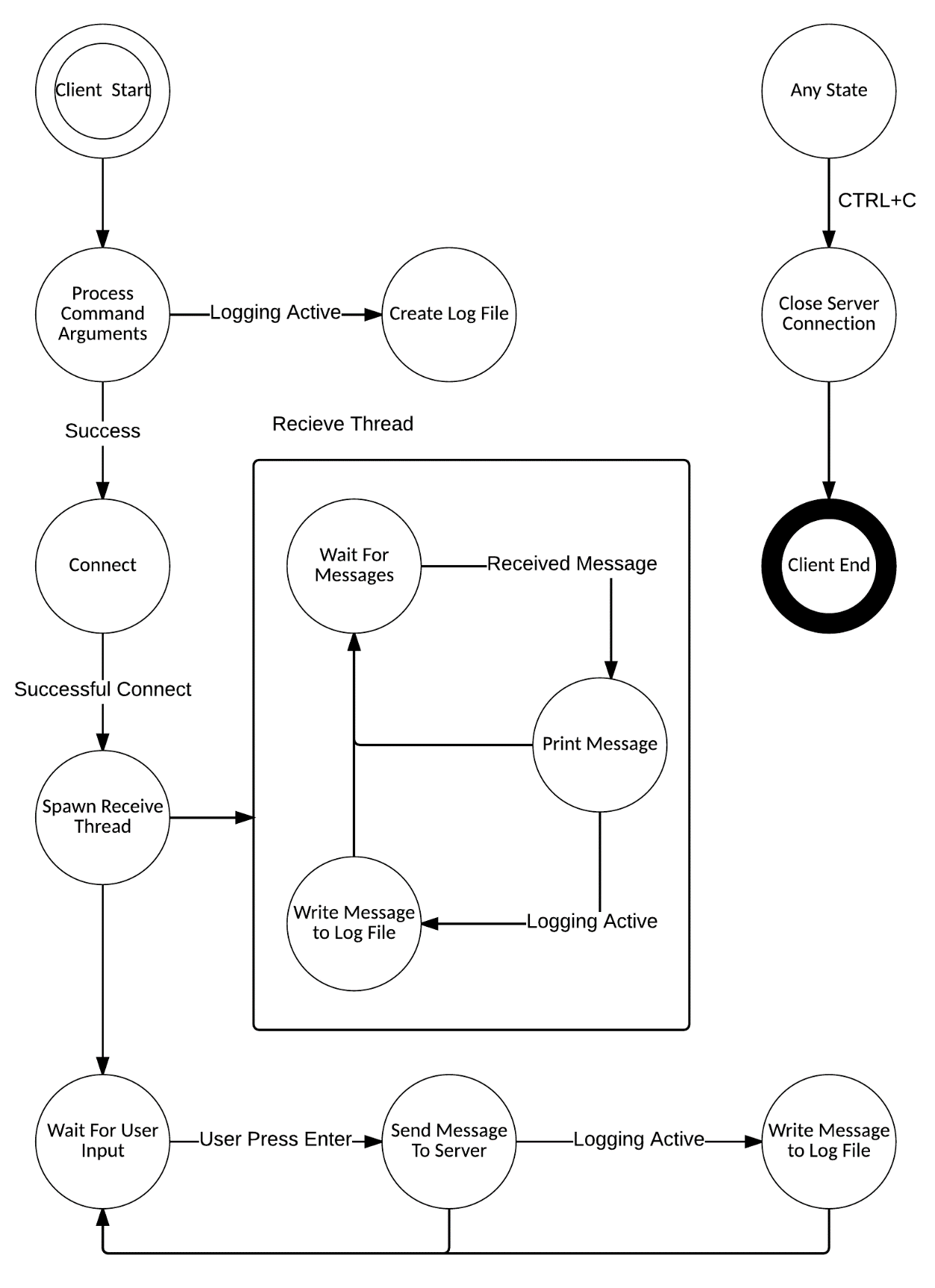
To execute it, run the following command in a terminal:

bash runtest.sh

# State Chart Diagram (Server)



# State Chart Diagram (Client)



# Pseudo Code

## Client

**Process Command Arguments**

if invalid host

return false

if invalid port

return false

assign username

if logactive

set global switch

return true

**Create Log File**

if logactive

open log file in append mode

**Connect**

create tcp socket

create sockaddr struct

initialize socketaddr members

create host pointer

connect call

go to Spawn Receive Thread

**Spawn Receive Thread**

spawn receive thread

go to Wait For User Input

**Receive Thread**

loop forever

loop on blocking recv call until buflen

if nothing read

exit program

if logactive

write received message to log

print out received message to console

**Wait For User Input**

loop forever

read in line from stdin to message string

append timestamp to message front

if username set

append username to message front

if (message length <= buffer length)

send message to server

write message to log file

else

print message too long error

**Any State**

if CTRL + C

go to Close Server Connection

**Close Server Connection**

close server socket descriptor

if logactive

close log file

program exit

# Pseudo Code (continued)

## Server

**Process Command Arguments**

if invalid port

print error message

exit program

go to Create & Initialize Socket

**Create & Initialize Socket**

create TCP listen socket

set socket options (reuse)

bind listen socket

listen for connections

initialize client socket desriptor array

go to Wait For Client Events

**Wait For Client Events**

select call

if FD\_ISSET on listen socket

go to Accept Connection

loop through client array

assign temp socket descriptor to current index in client array

if temp socket > 0 and FD\_ISSET

if receive message

go to Send Message To Other Clients

if 0 bytes received

go to Update Socket Descriptors

**Accept Connection**

accept new connection from listen socket

print new connection

update client array

set accepted socket

go to Wait For Client Events

**Send Message To Other Clients**

loop through client array

if FD\_ISSET and not sender

send recieved message

go to Wait For Client Events

**Update Socket Descriptors**

print connection closed message

close socket descriptor

FD\_CLR temp socket

update client array index

go to Wait For Client Events

**Any State**

if CTRL + C

go to Close Server Connection

**Close Open Connection**

close open sockets

program exit

# Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **DESCRIPTION** | **STEPS** | **EXPECTED RESULT** | **ACTUAL RESULT** | **RESULT** |
| 1 | Single client connects to the server | 1. Create server 2. Create Client 3. Send a message to the server | No echo on client side, server displays the message from client. | No echo on client side, server displays the message from client. | PASS |
| 2 | Three clients connect to the server | 1. Create server 2. Create Clients 3. Send a message from each client to the server | Each client sees the other clients’ messages. Server displays the messages from all the clients. | Each client sees the other clients’ messages. Server displays the messages from all the clients. | PASS |
| 3 | Client can reconnect to the server. | 1. Create the server 2. Create client 3. Client disconnects 4. Client connects again. 5. Client sends a message. | No echo on client side, server displays the message from client. | No echo on client side, server displays the message from client. | PASS |
| 4 | Disconnect the server with clients | 1. Create server 2. Create Clients 3. Disconnect server | The clients disconnect and the program exits with an error indicating the server has gone offline. | The clients disconnect and the program exits with an error indicating the server has gone offline. | PASS |
| 5 | Single client connects to the server with chat log enabled | 1. Create server 2. Create Client with log enabled 3. Send a message to the server | No echo on client side, server displays the message from client. The log file displays the message | No echo on client side, server displays the message from client. The log file displays the message | PASS |
| 6 | Multiple clients connect to the server with chat log enabled | 1. Create server 2. Create Clients 3. Send a message from each client to the server | Each client sees the other clients’ messages. Server displays the messages from all the clients. Log file displays the messages | Each client sees the other clients’ messages. Server displays the messages from all the clients. Log file displays the messages | PASS |
| 7 | Starting the Client/Server with wrong parameters | 1. Start Client/Server with incorrect parameters | Error message is displayed indicating proper command line arguments. | Error message is displayed indicating proper command line arguments. | PASS |
| 8 | Client Sends a message with a size greater than 511 | 1. Create server 2. Create Client 3. Send a long message to the server | Client prints an error indicating the maximum message size. | Client prints an error indicating the maximum message size. | PASS |

|  |  |
| --- | --- |
| TEST 1 | TEST 2 |
| TEST 3 | TEST 4  test4 |
| TEST 5 | TEST 6 |
| TEST 7  test7 | TEST 8  toolong |