# COMP 4981 – ASSIGNMENT 3 TESTING DOCUMENTATION

Yiaoping Shu + Tim Makimov

**Testing Documentation**

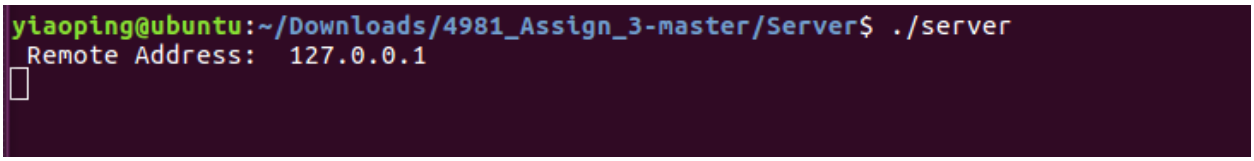
| Test # | Purpose | Steps | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | Able to open up the server | Open and run the server. | User is able to successfully run the server. | Passed |
| 2 | Able to open the client with proper IP and port | Open up the client and run the program, entering a good IP server and port number | User is able to run the program successfully. | Passed |
| 3 | User is able to type into the client program, entering any words | Run the client and attempt to type some letters. | User can type in the application | Passed |
| 4 | User can send data to the server | Enter any letters in the client, then press enter. | Cursor moves to the next line after pressing enter. | Passed |
| 5 | Other client can receive data from first client. | Open up two clients. Type any letters in first client, press enter, and check other client. | Words from client 1 appears on client 2's screen. | Passed |
| 6 | **Test for failure:** Typing in incorrect port and IP results in error | Type in incorrect IP and port when running the client. | An error appears displaying client cannot connect to server | Passed |
| 7 | Receiving client has timestamp displayed when data is sent to user. | Run two clients and the server. In client 1, send data. Check for correct data in client 2. Type in some data to send from client 2 to client 1. | Client 2 receives data from client 1 displayed with timestamp before message. Client 1 receives data with timestamp from 2. | Passed |
| 8 | Test client to client on | Open up client 1 on one computer. Open | Clients connect and run successfully, | Passed |

| | | | | |
|---|---|---|---|---|
| | different computers. | another client on second computer. Run a server on any of the computers and connect to server. Send data. | displaying data sent back and forth | |
| 9 | Ensure server receives connection on client connection | Open and run the server. With server on screen, run the client. | Server displays a “Remote Address: “ with the IP of the server displayed after. | Passed |
| 10 | Exiting the client results in correct message | Run the client and press ctrl-c. | Client should exit gracefully with no errors. | Passed |
| 11 | Exit the server result in correct action. | Run the server with clients connected and press ctrl-c. | Both server and clients exit gracefully with no errors, displaying server closed connection message on client. | Passed |
| 12 | Server able to handle loads of data. | Send more than 10 messages back and forth between clients. | Clients can receive data from each other with no errors or delays. | Passed |
| 13 | Client able to save chat log to a file | Specify file name as 4th argument | File is created. File log contains all sent and received messages. | Passed |
| 14 | Select function is handling file descriptors correctly | 1. User connected<br>2. User disconnected<br>3. User sent message | 1. New fd added to set<br>2. Fd removed from set<br>3. Message echoed back to all users except for the sender | Passed |

**Testing Screenshots**

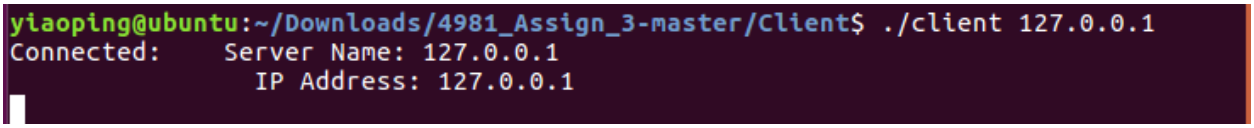
**Test #1**

Able to open up the server



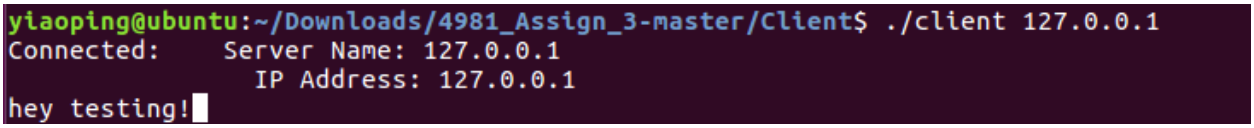

**Test #2**

Able to open the client with proper IP and port



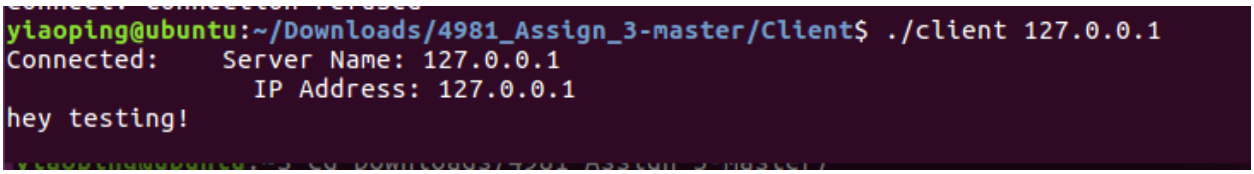

**Test #3**

User is able to type into the client program, entering any words



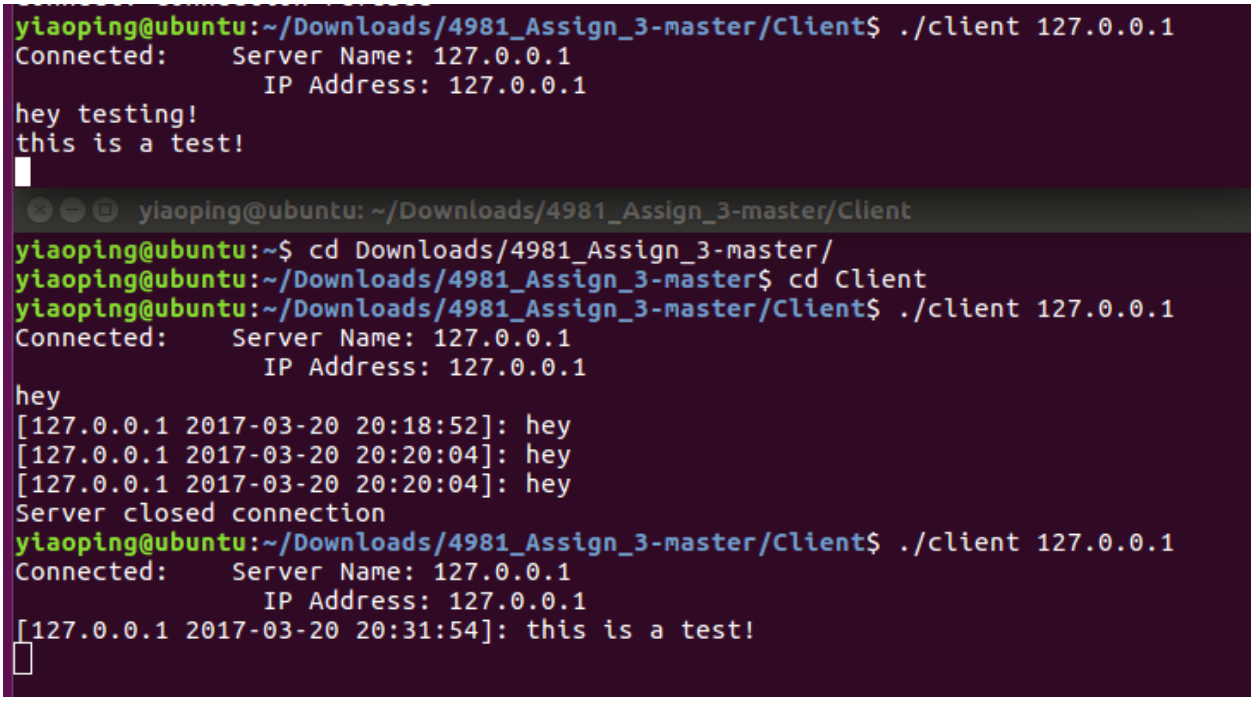

**Test #4**

User can send data to the server



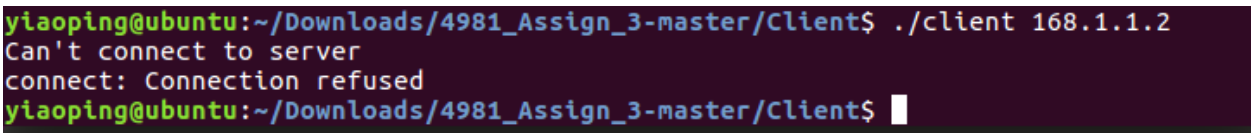
**Test #5**

Other client can receive data from first client



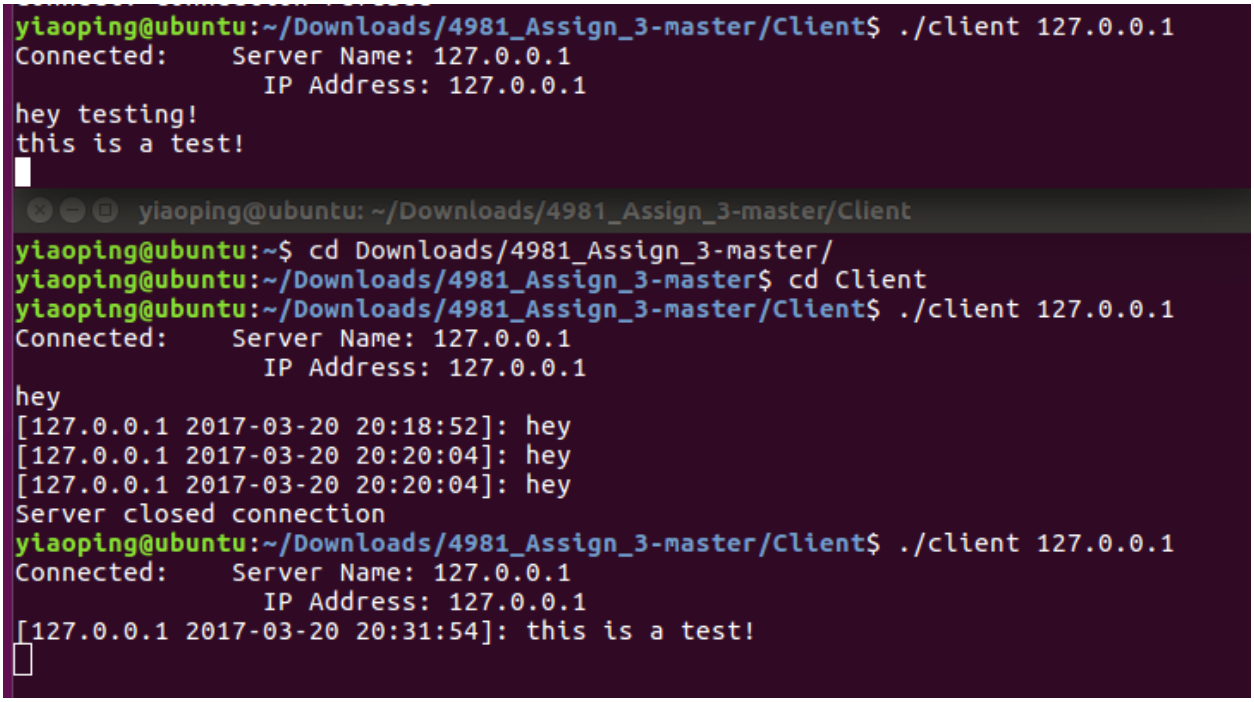

**Test #6**

**Test for failure:** Typing in incorrect port and IP results in error



**Test #7**

Receiving client has timestamp displayed when data is sent to user.

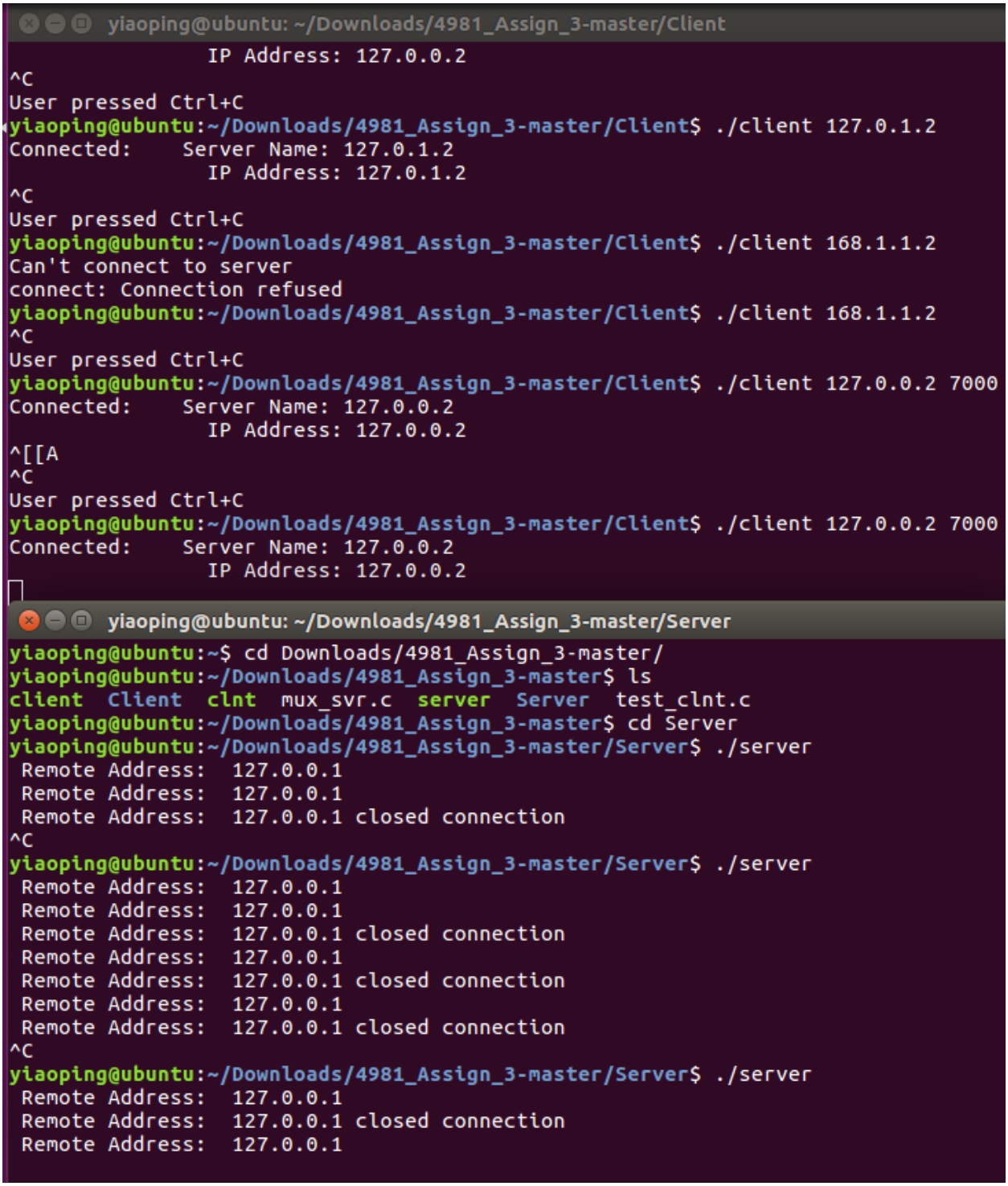



**Test #8**

Test client to client on different computers.

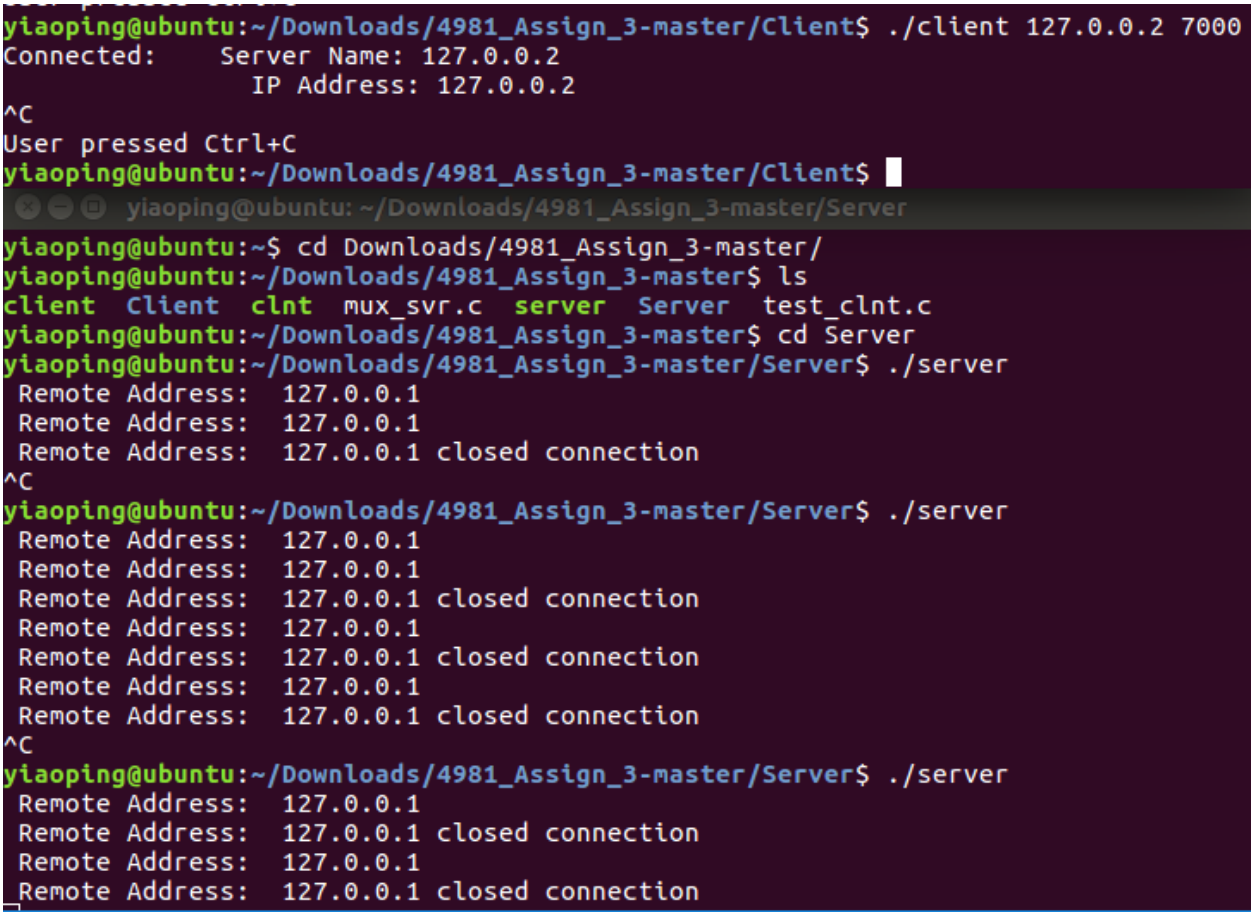
**Test #9**

Ensure server receives connection on client connection



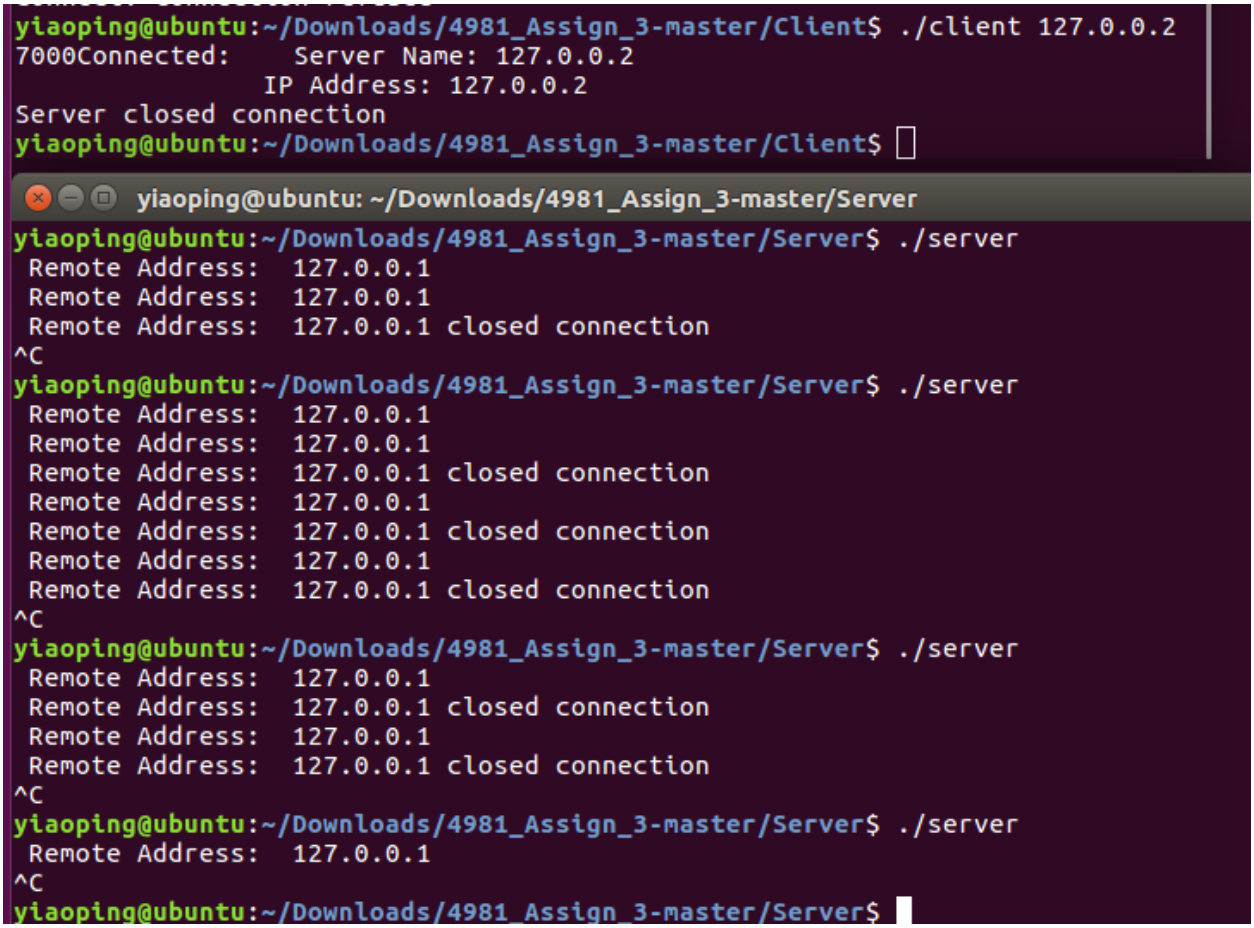
**Test #10**

Exiting the client results in correct message



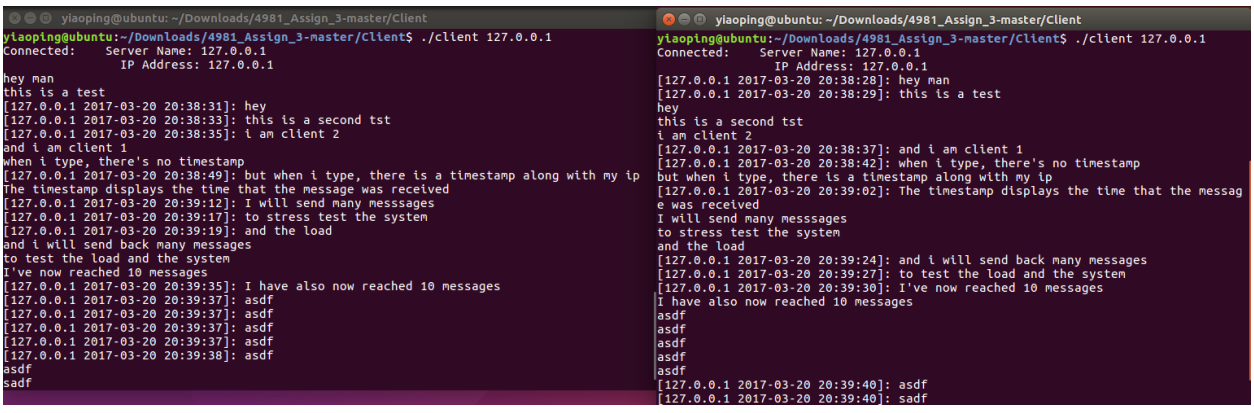
**Test #11**

Exit the server results in correct action.




**Test #12**

Server able to handle loads of data.



**Test #13**

Specify filename as 4th argument and connect

```
[megapers@localhost Client]$ ./client localhost 7000 Tim test1.txt
Connected!
Server Name: localhost
IP Address: 127.0.0.1
```

File is created

```
[megapers@localhost Client]$ ls
client    clientConnect.c  client.h  clientSelect.c  Makefile   test.txt
client.c  clientConnect.o  client.o  clientSelect.o  test1.txt
[megapers@localhost Client]$
```

Chat conversation is saved into file log

```
[Aman 2017-03-22 14:35:04]: Hello class!
[Tim 2017-03-22 14:35:13]: Hello!
[Yiaoping 2017-03-22 14:35:36]: Good morning!
[Aman 2017-03-22 14:37:01]: Time to check your Assignment 3...
[Yiaoping 2017-03-22 14:37:50]: Uh-oh! We didn't implement QT :..(
[Tim 2017-03-22 14:38:10]: But we can save file logs!!!
```

**Test #14**

Users connected

```
select(6, [3 4 5], NULL, NULL, NULL)    = 1 (in [3])
accept(3, {sa_family=AF_INET, sin_port=htons(34562), sin_addr=inet_addr("1
.1")}, [16]) = 6
write(1, " Remote Address:  127.0.0.1\n", 28 Remote Address:  127.0.0.1
) = 28
select(7, [3 4 5 6], NULL, NULL, NULL
```

File descriptors of connected users

User disconnected

```
select(7, [3 4 5 6], NULL, NULL, NULL)  = 1 (in [4])
recvfrom(4, "", 255, 0, NULL, NULL)     = 0
write(1, " Remote Address:  127.0.0.1 clos"..., 46 Remote Address:  127.0.0.1 cl
osed connection
) = 46
close(4)                                = 0
select(7, [3 5 6], NULL, NULL, NULL
```

Client with fd 4 closed connection to server

User sent message

```
select(7, [3 4 5 6], NULL, NULL, NULL)  = 1 (in [4])
recvfrom(4, "[Tim 2017-03-22 14:56:28]: Hello"..., 255, 0, NULL, NULL) = 2
sendto(5, "[Tim 2017-03-22 14:56:28]: Hello"..., 255, 0, NULL, 0) = 255
sendto(6, "[Tim 2017-03-22 14:56:28]: Hello"..., 255, 0, NULL, 0) = 255
select(7, [3 4 5 6], NULL, NULL, NULL
```

Message received from client with fd 4.

Message only sent to clients with fd 5 and 6.

After sending, get back to select (monitor fd set).