**Documentation**

**Client**

**Server**

The server has many tasks that have to be dealt with:

1) Listen for connections

2) Open a new thread per connection

3) Store all the shared data in a Mailbox System

4) Make sure that no threads are trying to read/write to shared memory simultaneously

5) Listen for the ctrl-c signal to properly free all memory and shut down the server connection

Listening for client connections was put in the main function because accept() will block until a connection is established. When a connection occurred, a thread will run with an initClient() function that had all the code to properly allow the client to read/write data to the Mailbox System.

**Data Structures used for Mailbox System**

The Mailbox System is a linked list of Message Box nodes. A Message Box node consists of 4 parts: The name of the message box, whether the message box is open or not, a pointer to the next message box, and also a pointer to a Queue. The queue is a linked list of nodes, where each node holds a character array to hold a message, and each node points to a next node. The queue system is used to store messages from the PUTMG and NXTMG commands in the proper order.

**Handling Synchronization and Freeing Memory**

Mutex’s are used in order to make sure that no two clients were reading or writing information to the Mailbox System at the same time. When the user presses ctrl-c to end the server, the signal is detected and calls an exitServer() function. All threads, mutexes, and the entire Mailbox System gets free()’d properly to ensure no memory leaks are present.

**Test Case 1**

In this test case we wanted to test for general functionality of creating, opening, putting, and getting a message from a message box.

**Client**

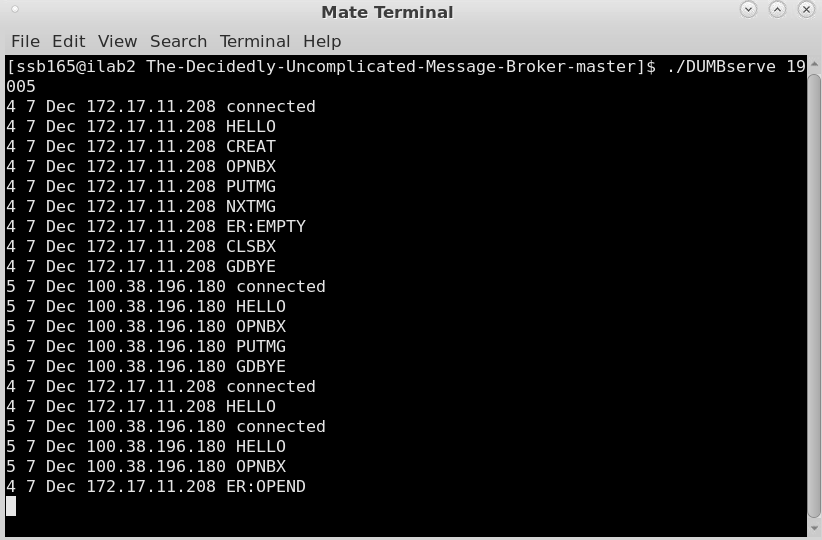
./DUMBclient [ilab2.cs.rutgers.edu](http://ilab2.cs.rutgers.edu/) 19005  
Success. You are connected.  
Please choose a command from the command list below:  
1. quit  
2. create  
3. delete  
4. open  
5. close  
6. next  
7. put  
create  
Okay, enter the name of the box  
create:> messagebox1  
Successfully created  
open  
Okay, open which message box?  
open:> messagebox1  
Successfully opened  
put  
Okay, enter the message you want to put in the box.  
put:> hello, there!  
Success. Message was put in the message box  
next  
hello, there!  
next  
Error. No messages left in this message box  
close  
Okay, close which message box?  
messagebox1  
Successfully closed  
quit  
Connection closed

**Server**

4 7 Dec 172.17.11.208 connected  
4 7 Dec 172.17.11.208 HELLO  
4 7 Dec 172.17.11.208 CREAT  
4 7 Dec 172.17.11.208 OPNBX  
4 7 Dec 172.17.11.208 PUTMG  
4 7 Dec 172.17.11.208 NXTMG  
4 7 Dec 172.17.11.208 ER:EMPTY  
4 7 Dec 172.17.11.208 CLSBX  
4 7 Dec 172.17.11.208 GDBYE

**Test Case 2**

In this test case, we wanted to test multi-connections within our server system. We repeated Test Case 1 on 2 clients, one run on linux, the other run on Windows. We creating a messagebox1 and opened messagebox1 on Linux. The Windows client then tried to open messagebox1, but got an ER:OPEND, because the Linux client already opened it first and did not close it. The screenshot of the server demonstrates these connections by looking at what the server outputted on the next page.



**Test Case 3**

With this test case we wanted to test the limit for putting a really really long message into a message box.

**Client**

open

Okay, open which message box?

open:> messagebox1

Successfully opened

put

Okay, enter the message you want to put in the box.

put:> kmplikhhpirbqnkxuiyjcqfpriddyertovsaxvlrfacmtvisrjjmphrvgwgnlyfwifjtuzhzcwperpcfrosfvobgrugdauxirlhdirbgxdcxvlvfhsqgzaaylmnqfcofbzaddyxohjiznujwyconvqsigjqcdvwvgqrtcppllmefqrkzwrfvstqjftatvltldxoagukmsrqrcvekoapipiihacsvbidtdlbjqojzihnnaxtxiejujkxinnzuvxcloxztxilcvregtmjnvastckjycuveeiugxnbstsdsbechlmzjojrprteedrrnnanzpltstdjxreuoqfjbbauwmxulfkpjgptkwpuxbkjdmmrymanrkjwqlvnrlvlbpgelgdhyuiyzscimupyvenclhqajchkzrpyosghostabvjmyppgrlxpnkmlasiedngkoopoodlknuzzvotwyrpkabsyuygjaxivmjulkuubihjvqgjuecdwlquunkvieahpaoquakczsopoocooqrneacvuubtcdsvjbogwbbzqnezjqjfihryzkeqvkbhxoaoorjxhwzahkasldskqgmhqjgvpaqpmfycmsajazizbbhcbidjlzpjmuiszmurqnowywpwacxclhigsjdvccdynuzyjncrhvqlyjhtedaiewgybkmiyjtonprtuvnafagwpbdsjqsstvqisyxjqrlyscvinnfohtavqdocijsovabvtgzturppygyplqojcfdywmbakxmgytpwdfoiydpctnnrwerksmnqfc

Your message was trimmed

Success. Message was put in the message box

next

hey, world

next

kmplikhhpirbqnkxuiyjcqfpriddyertovsaxvlrfacmtvisrjjmphrvgwgnlyfwifjtuzhzcwperpcfrosfvobgrugdauxirlhdirbgxdcxvlvfhsqgzaaylmnqfcofbzaddyxohjiznujwyconvqsigjqcdvwvgqrtcppllmefqrkzwrfvstqjftatvltldxoagukmsrqrcvekoapipiihacsvbidtdlbjqojzihnnaxtxiejujkxinnzuvxc

Here we see that the user inputted an 800 character long string, however the client notifies the user that their message has been trimmed and saved. When the user asks for the message back, they get 255 characters. This is because the limit of a message is 256 bytes including the null terminator, 255 bytes free to set by the user.

**Test Case 4**

This test case was created to make sure that 2 clients cannot try to open/delete/close each other’s message boxes. The client output is pasted below for client 1. Note that client 2 has messagebox2 open and messagebox3 does not exist.

**Client**

open  
Okay, open which message box?  
open:> messagebox1  
Successfully opened  
close  
Okay, close which message box?  
messagebox2  
Error. This box is not currently open  
close  
Okay, close which message box?  
messagebox3  
Error. This box is not currently open  
delete  
Okay, enter the name of the box  
messagebox1  
Error. This box is currently open  
delete  
Okay, enter the name of the box  
messagebox3  
Error. Box doesn't exist  
close  
Okay, close which message box?  
messagebox2  
Error. This box is not currently open  
close  
Okay, close which message box?  
messagebox1  
Successfully closed  
delete  
Okay, enter the name of the box  
messagebox1  
Successfully deleted  
delete  
Okay, enter the name of the box  
messagebox2  
Error. This box is currently open