Report for Assignment 1 : CS3205

Simple Client-Server Implementation using Socket Programming

Author : Shreesha G Bhat (CS18B103)

Basic Design of the Code for the Client and Server

Design of emailserver.c

There are two components, the network_interface and the command processor.

The command processor

The command processor is broken up into many functions such as do_delm(), do_user(), do_send() etc.

The network interface

Has an infinite loop, which reads messages from the client, processes them and writes messages back to the client. The processing involves parsing the received messages, invoking the required functions from the command processor, building the message to be sent back to the client.

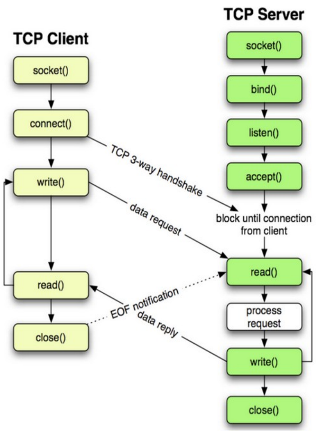
Design of emailclient.c

There is one component. The network_interface (a merger of the user shell and the component that actually sends messages)

The network interface

Has an infinite while loop (like a shell) which reads inputs from the user. Parses the inputs and sends the messages to the server. On receiving the messages back from the server, it parses them and displays the same to the user.

Apart from the respective components, both the C programs have a main program which invokes the socket library functions to set-up a client-server TCP/IP network socket connection.



Extra features implemented

1) Introduced Passwords

The Adduser and SetUser commands have been modified to include a username and password based authentication. This required maintaining the necessary state and files at the server end.

Explanation :

The Adduser command is invoked as

Adduser <User_ID> <Password>

provided the user does not already exist, the password and user_id are sent to the server as a message (plaintext i.e. no encryption of any kind is performed) and these are stored in the state of the server (in files as well as global variables).

The SetUser command is then invoked as

SetUser <User_ID> <Password>

Again, the password and user_id are sent to the server as a message. The server authenticates the password by string comparing it with the password sent earlier (which was stored) and sends the required acknowledgement back.

Note :

This is in no way a secure implementation as primarily, the password are visible on the terminal when being inputted. They are also sent as plaintext over a network which is never safe. They are also stored on the server end without any encryption in global variables and in files.

But this implementation gives a very basic idea as to how username and password based authentication is performed.

2) Emulated the notion of a server failure and restoring of server state on restart

If the server program stops running abruptly (can be emulated using a Ctrl-C) and the MAILSERVER directory is kept intact. The server program can be restarted and some state is effectively restored.

This is implemented as follows,

There is an init_server function defined which is invoked in main. This function first checks if a MAILSERVER directory is present (which contains a file with the usernames and passwords, along with the email spool files from a prior run). In which case, it reads the stored file containing the username and passwords and restores the global variables containing the user_id information and password information.

If the MAILSERVER directory is absent, then the init_server function creates the same along with a new file for the usernames and passwords.

Note :

Again, this is in no way a secure/efficient implementation, but reflects the general idea. In a real life case, more state variables and context variables could be restored.

3) Added “help” functionality to print out the commands and their uses

Putting command “help” in the Main Prompt or the user prompt prints out the commands available and their uses. The implementation is a very simple addition to the user shell code.

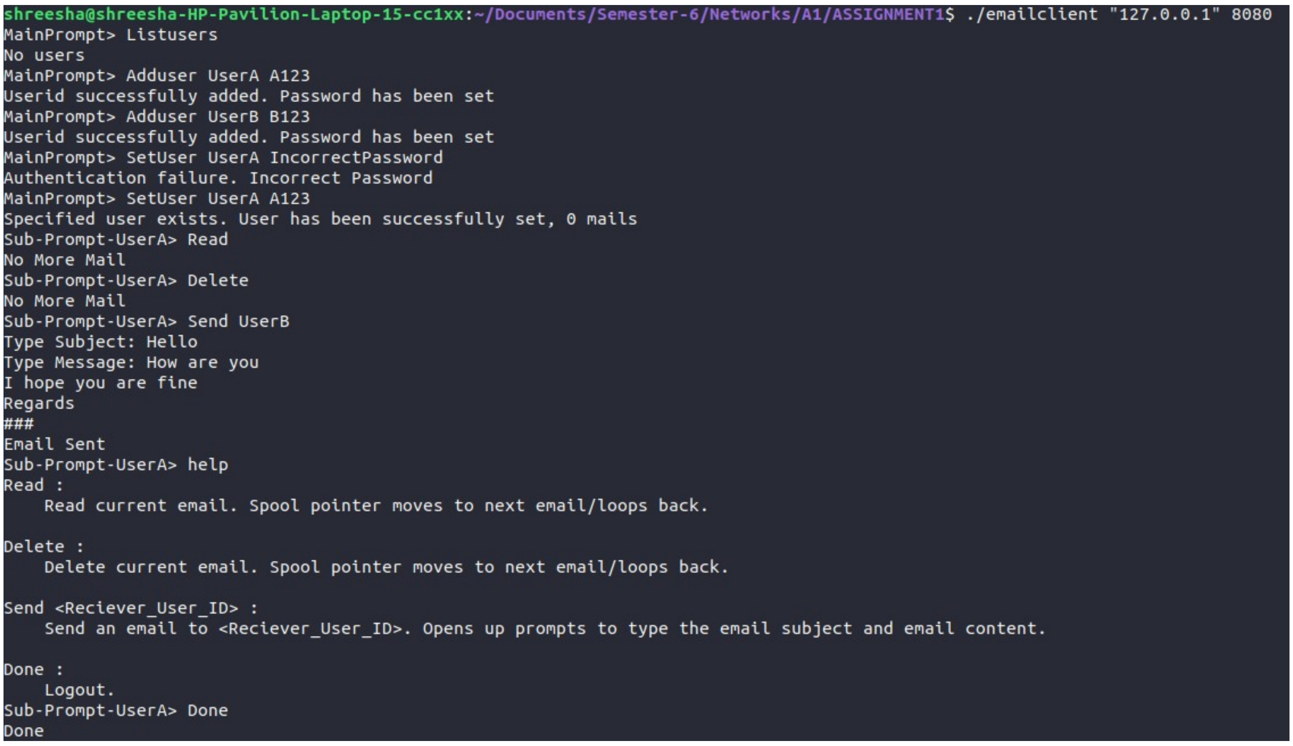
Also added functionality to give Invalid Command messages in case of invalid commands being input.

Learning Objectives : How was the lab helpful?

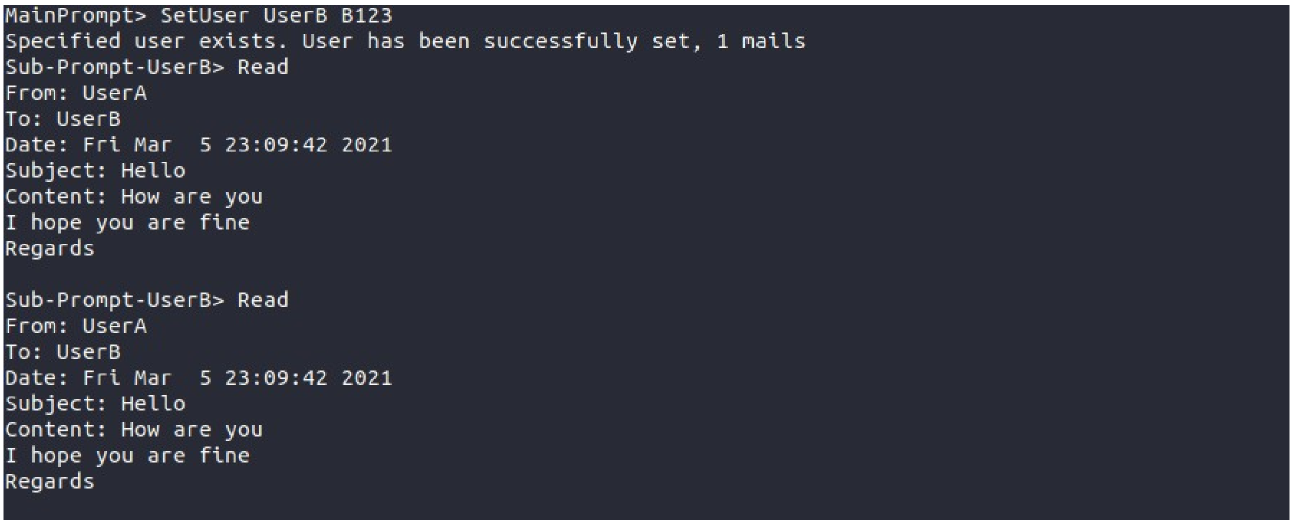
- Primarily, I learnt about socket programming for the first time. Helped me get familiar with the various functions of the socket library and how to set-up a connection for a simple client-server message passing mechanism.
- I also learnt about the hardships and subtleties involved in maintaining and building a server. This involves system programming in C, working with Files and Directories, working with arrays, strings etc. I also learnt how to implement a simple user-shell like interface while writing the client side code.
- I also got insight into writing and maintaining a large piece of code over a period of time (duration of the assignment), such as writing useful comments, abstracting away frequently used components into functions and writing code that is fairly easy to read and understand.
- This lab helped me develop a lot of interest in the further assignments and got me motivated to understand and work with Networks.

Screenshots Showing some basic features

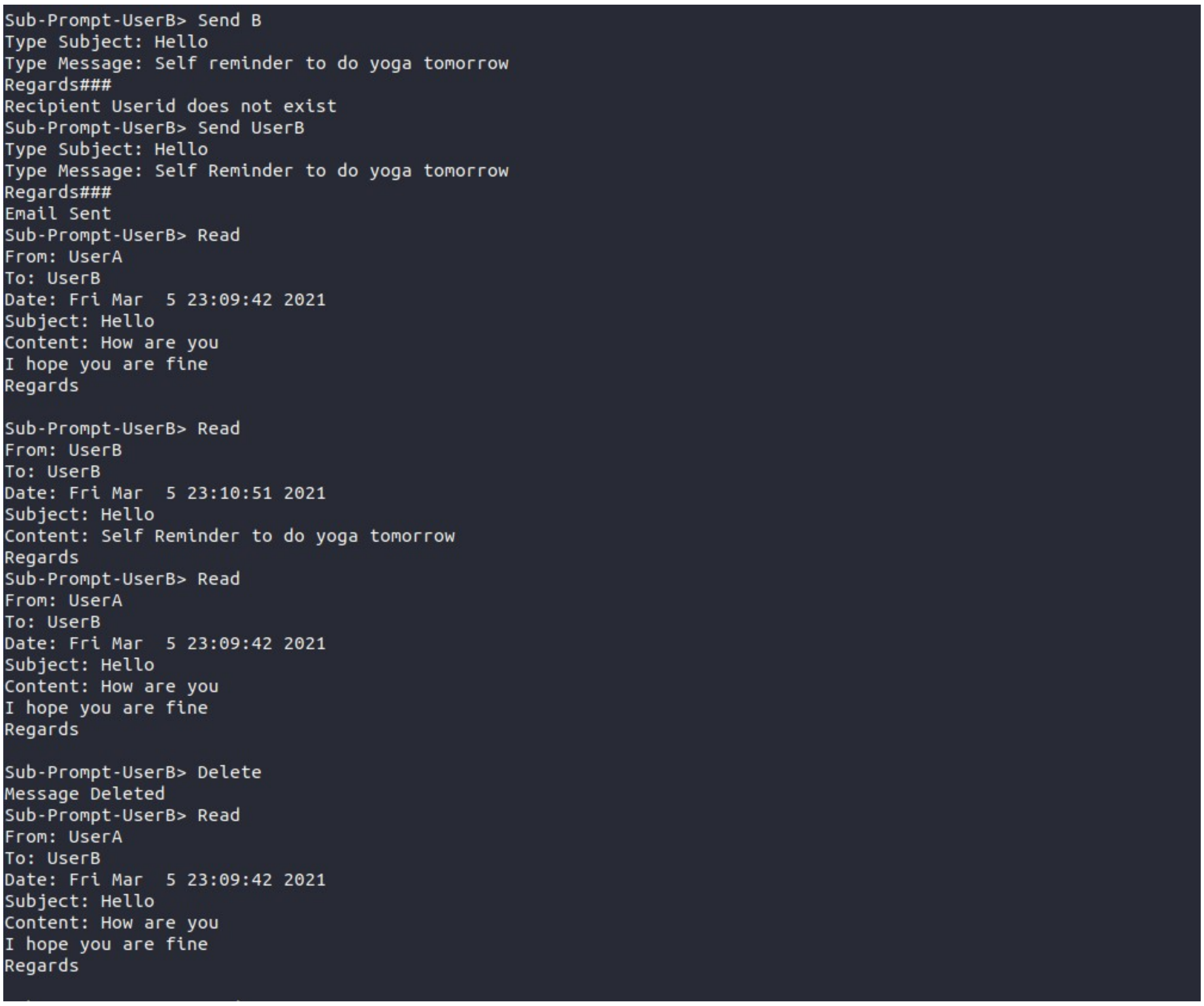
The below 4 screenshots show basic features such as Password Authentication, Sending mails to a different user, Self emails, Loopback on Reads and Loopbacks on Delete, the help feature etc.



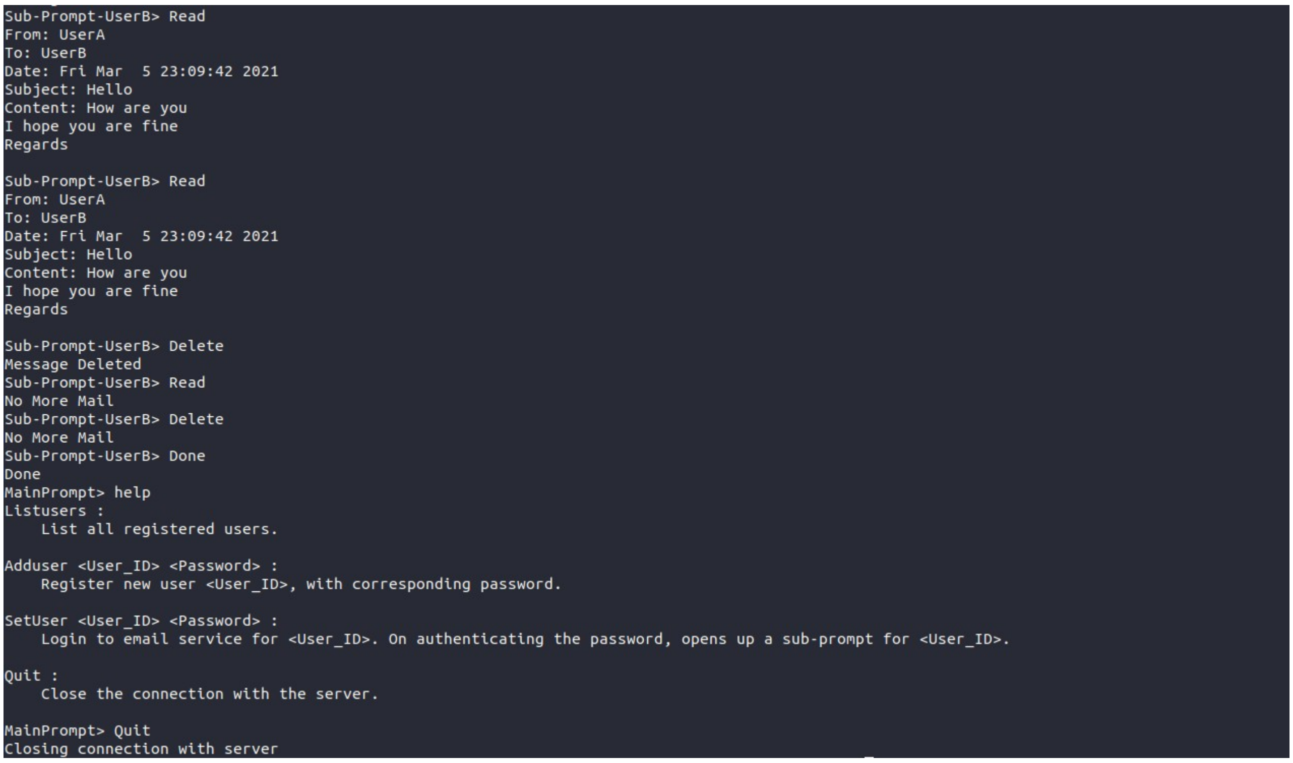
```
shreesha@shreesha-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Semester-6/Networks/A1/ASSIGNMENT1$ ./emailclient "127.0.0.1" 8080
MainPrompt> Listusers
No users
MainPrompt> Adduser UserA A123
Userid successfully added. Password has been set
MainPrompt> Adduser UserB B123
Userid successfully added. Password has been set
MainPrompt> SetUser UserA IncorrectPassword
Authentication failure. Incorrect Password
MainPrompt> SetUser UserA A123
Specified user exists. User has been successfully set, 0 mails
Sub-Prompt-UserA> Read
No More Mail
Sub-Prompt-UserA> Delete
No More Mail
Sub-Prompt-UserA> Send UserB
Type Subject: Hello
Type Message: How are you
I hope you are fine
Regards
###
Email Sent
Sub-Prompt-UserA> help
Read :
    Read current email. Spool pointer moves to next email/loops back.

Delete :
    Delete current email. Spool pointer moves to next email/loops back.

Send <Reciever_User_ID> :
    Send an email to <Reciever_User_ID>. Opens up prompts to type the email subject and email content.

Done :
    Logout.
Sub-Prompt-UserA> Done
Done
```



```
MainPrompt> SetUser UserB B123
Specified user exists. User has been successfully set, 1 mails
Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards

Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards
```

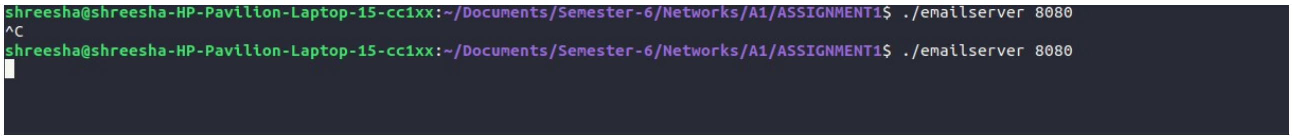


```
Sub-Prompt-UserB> Send B
Type Subject: Hello
Type Message: Self reminder to do yoga tomorrow
Regards###
Recipient Userid does not exist
Sub-Prompt-UserB> Send UserB
Type Subject: Hello
Type Message: Self Reminder to do yoga tomorrow
Regards###
Email Sent
Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards

Sub-Prompt-UserB> Read
From: UserB
To: UserB
Date: Fri Mar  5 23:10:51 2021
Subject: Hello
Content: Self Reminder to do yoga tomorrow
Regards
Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards

Sub-Prompt-UserB> Delete
Message Deleted
Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards
```

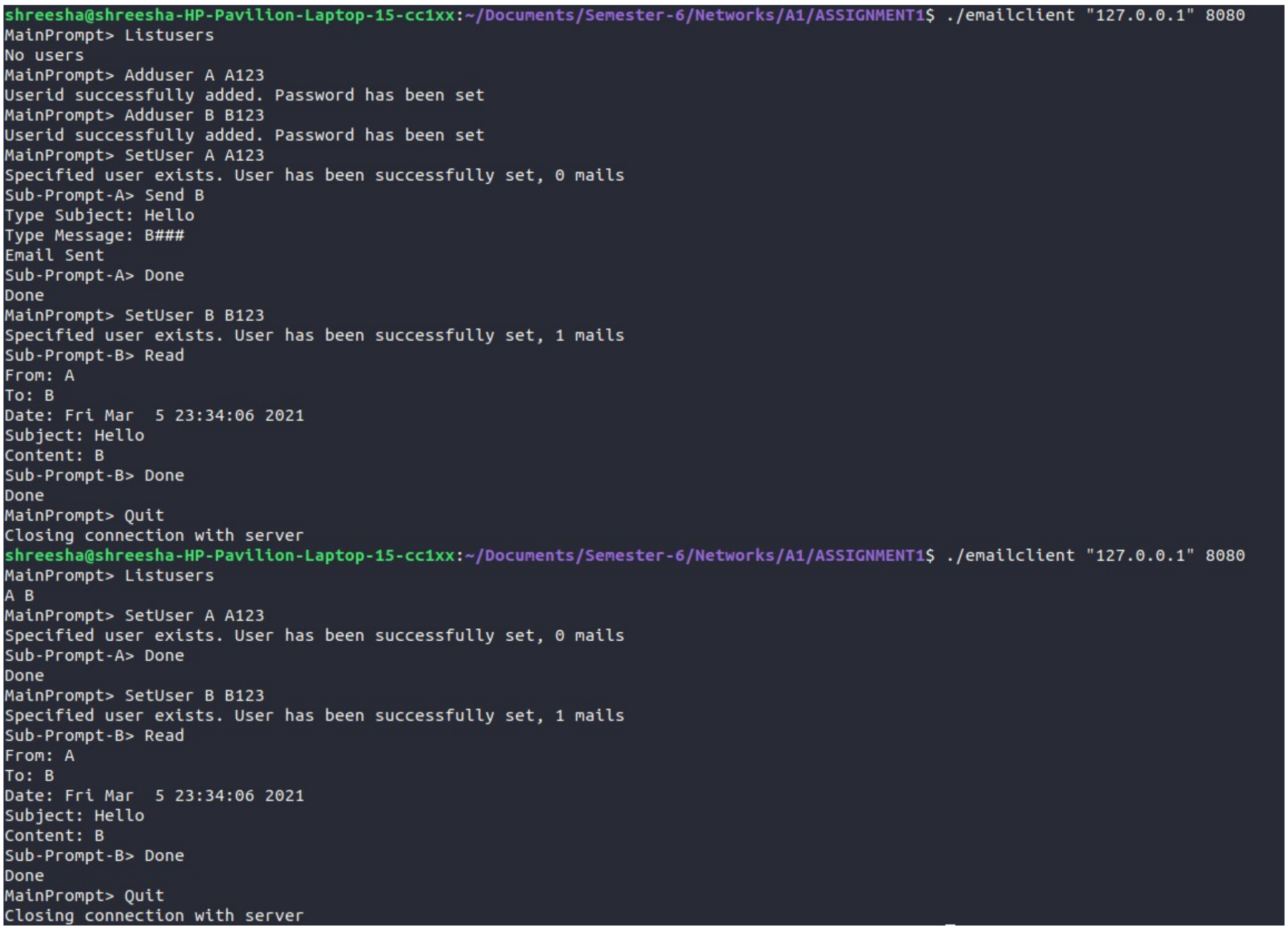


```
Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards

Sub-Prompt-UserB> Read
From: UserA
To: UserB
Date: Fri Mar  5 23:09:42 2021
Subject: Hello
Content: How are you
I hope you are fine
Regards

Sub-Prompt-UserB> Delete
Message Deleted
Sub-Prompt-UserB> Read
No More Mail
Sub-Prompt-UserB> Delete
No More Mail
Sub-Prompt-UserB> Done
Done
MainPrompt> help
Listusers :
    List all registered users.

Adduser <User_ID> <Password> :
    Register new user <User_ID>, with corresponding password.

SetUser <User_ID> <Password> :
    Login to email service for <User_ID>. On authenticating the password, opens up a sub-prompt for <User_ID>.

Quit :
    Close the connection with the server.

MainPrompt> Quit
Closing connection with server
```

Screenshots showing the Server stopping (Ctrl-C) and restoring of state at the client side.



```
shreesha@shreesha-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Semester-6/Networks/A1/ASSIGNMENT1$ ./emailserver 8080
^C
shreesha@shreesha-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Semester-6/Networks/A1/ASSIGNMENT1$ ./emailserver 8080
```



```
shreesha@shreesha-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Semester-6/Networks/A1/ASSIGNMENT1$ ./emailclient "127.0.0.1" 8080
MainPrompt> Listusers
No users
MainPrompt> Adduser A A123
Userid successfully added. Password has been set
MainPrompt> Adduser B B123
Userid successfully added. Password has been set
MainPrompt> SetUser A A123
Specified user exists. User has been successfully set, 0 mails
Sub-Prompt-A> Send B
Type Subject: Hello
Type Message: B###
Email Sent
Sub-Prompt-A> Done
Done
MainPrompt> SetUser B B123
Specified user exists. User has been successfully set, 1 mails
Sub-Prompt-B> Read
From: A
To: B
Date: Fri Mar  5 23:34:06 2021
Subject: Hello
Content: B
Sub-Prompt-B> Done
Done
MainPrompt> Quit
Closing connection with server
shreesha@shreesha-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Semester-6/Networks/A1/ASSIGNMENT1$ ./emailclient "127.0.0.1" 8080
MainPrompt> Listusers
A B
MainPrompt> SetUser A A123
Specified user exists. User has been successfully set, 0 mails
Sub-Prompt-A> Done
Done
MainPrompt> SetUser B B123
Specified user exists. User has been successfully set, 1 mails
Sub-Prompt-B> Read
From: A
To: B
Date: Fri Mar  5 23:34:06 2021
Subject: Hello
Content: B
Sub-Prompt-B> Done
Done
MainPrompt> Quit
Closing connection with server
```