Ben Sterenson 304745375 bensterenson@gmail.com
Michael Palarya 309417418 palarya@gmail.com

Communication Networks Ex1

Protocol Definition:

Server-Client protocol:

Header of message			Body of message	
length	opcode	status	parameter1	..parameterN

Header structure:

- Total length of message expected to read
- Opcode describing the type of message sent
- Status byte indicating success/fail on server side

Body structure:

- Message fields are separated by a special '\n' delimiter byte

Example:
"0000000006ls" - length of message is 6 bytes, L = Login request, and S = success

Server-Client protocol:

Header of message	Body of message

Header structure:

- Total length of message expected to read
- Opcode describing the type of message sent

Body structure:

- Message fields are separated by a special '\n' delimiter byte

Example:
"0000000012lben\n123" - length of message is 12 bytes, L = login request, with username=ben and password=123 - separated by a '\n' delimiter.

Opcodes:

- SHOW_INBOX = 'i'
- GET_MAIL = 'm'
- DELETE_MAIL = 'd'
- QUIT = 'q'
- COMPOSE = 'c'
- LOGIN = 'l'
- SERVER_SUCCESS = 's'
- SERVER_FAILED = 'f'

Program description:

The program is a mail server implementing a TCP protocol communication, written in C++.
The mail server was built in such a way that it can handle multiple clients asynchronously.

The following commands are supported by the mail server:

SHOW_INBOX - will show the inbox of the logged user. Output:

#Mail_id Sender Subject

GET_MAIL mail_id - gives the ability to view a specific mail based on given mail_id. Output:

To:
From:
Subject:
Text:

DELETE_MAIL mail_id - gives the ability to delete a specific mail based on given mail_id.

COMPOSE - gives the ability to compose a new mail and send it to a user.

Format:

From:
To: (supports multiple users Format: user1,user2…)
Subject:
Text:

LOGIN (hidden command) - User should enter his credentials in order to log into his mailbox.

Format:

User: username
Password: password

QUIT - Closes the connection between client and the server.

How to run the program:

- Compile the files using Server and Client side (Command: make all \ make clean)
- Run server side: ./mail_server users_file [port]
    - User_file should be a path to a text file containing Users & Passwords tab-delimited.
    - Port is optional - default port is 6423
- Run Client side: ./mail_client [hostname [port]]
    - Hostname - can be a IP address\name(localhost)
    - Port - is optional - default port is 6423
    - Beware you cannot provide port without hostname.
- Once client is connected to Server client will present a welcome message:
  "Welcome! I am simple-mail-server."
- Enter user credentials in order to enter user's mailbox Format should be as following: Format:
  User: username
  Password: password
- If username & password are correct, the client will print 'Connected to server'.
- Once you are connected to server you can use any of the following commands as explained above:
  SHOW_INBOX
  GET_MAIL
  DELETE_MAIL
  COMPOSE
  Quit.

Program Structure:
The program was written in C++ and makes use of object oriented design.

Server project files:

.cpp source file	**.h header file**	**notes**
Mail_server.cpp		main function
	Mail.h	Struct SMail
Sever.cpp	Server.h	Singleton Class CServer

Client project files:

.cpp source file	**.h header file**	**notes**
Client.cpp	Client.h	Class CClient
main.cpp		main function

Common files for both projects:

.cpp source file	**.h header file**	**notes**
General.cpp	General.cpp	Common functions and defs
Message.cpp	Message.h	Class CMessage
Reply.cpp	Reply.h	Class CReply
MSG_Login.cpp	MSG_Login.h	Specific message structure, parsing, executing and replying methodologies.
MSG_Compose.cpp	MSG_Compose.h	
MSG_Delete_Mail.cpp	MSG_Delete_Mail.h	
MSG_Show_Inbox.cpp	MSG_Show_Inbox.h	
MSG_Get_Mail.cpp	MSG_Get_Mail.h	
MSG_Quit.cpp	MSG_Quit.h	