Mail Server Task Specification Doc

Students

Name	Last Name	TAU Username	ID
Liron	Gazit	lironemilyg	305774382
Lior	Shneider	shneider2	302814355
Matan	Gizunterman	gizunterman	303157804

**All use TAU Emails account with TAU username

Readme

- The Mail-Server protocol is defined at the next page.
- In order to use the mail server each client should build applications that implement the given protocol.

- In our local use case we use a client application, written in C, that implements the server protocol.

- Testing and using our client and server applications - follow the next instructions:

1. Extract all files includes in the archived (.zip) file into some directory (Linux OS).
2. Open two separated Linux terminals and go to the extracted files path.
3. Run the make file included by using the ***"make"*** command.
4. Run the mail_server application with the user.txt files as argument.
   in order to use customized port, pass it in the second argument, otherwise the mail server will use the default port 6432.
   "./mail_server ./users.txt 9999"
5. Now go to the second Linux terminal in order to run the client application and connect to the server.
6. Run the mail_client application ***"./mail_client".*** With default port 6432.
   in order to use customized host address or port number, pass the wanted values in the first and second arguments:
   "./mail_client" 127.0.0.1 9999"
7. Now, if no errors occurred, you can enter you username and password and authenticate with the server:
   "User: username_value"
   "Password: password_value"
8. At this moment, if the client authenticated successfully, you can use your email basic command (***SHOW_INBOX, COMPOSE, GET_MAIL, DELETE_MAIL, QUIT***) as specified in the task instructions.
9. At any time, the client can enter S_QUIT command and shutdown itself and the mail_server application

Server Side Protocol

1. Authentication

Authenticate the user with the system and obtain the auth_token

Input Message

Type	Data
String	[opCode][data_size][username_value][tab][password_value] “00013john doe”

Data Values

Name	Size	Value
opCode Size username Delimiter password	Byte 4 x Bytes Dynamic 1 x Bytes Dynamic	1 Size of total data received Client username value needed to authenticate space Separator between values Client password value needed to authenticate

Response

Type	Response
String String	“Connected to server” “failed to connect to server”

2. SHOW INBOX

Get the new updates

Input Message

Type	Data
String	[opCode][data_size] “10006”

Data Values

Name	Size	Value
opCode Size	Byte 4 x Bytes	1 Size of total data received

Response

Type	Response
String String	All Mails Inbox relative to the connected user “No Mails to show in Inbox”

3. GET MAIL

Input Message

Type	Data
String	[opCode][data_size][mail_id] “200105”

Data Values

Name	Size	Value
opCode Size mail_id	Byte 4 x Bytes dynamic	1 Size of total data received Id of connected user specific mail

Response

Type	Response
String String	The requested mail Content “The requested mail does not exist in DB”

4. DELETE MAIL

Input Message

Type	Data
String	[opCode][data_size][mail_id] “3john do”

Data Values

Name	Size	Value
opCode Size mail_id	Byte 4 x Bytes dynamic	1 Size of total data received Id of connected user specific mail

Response

Type	Response
	No response from server

5. COMPOSE

Input Message

Type	Data
String	[opCode][size][username_value][tab][password_value] “40100To: user1,user2\nSubject: someSubject\nText:someContent\n”

Data Values

Name	Size	Value
opCode Size mail_id	Byte 4 x Bytes dynamic	1 Size of total data received Id of connected user specific mail

Response

Type	Response
	No response from server

4. QUIT

Input Message

Type	Data
`String`	`[opCode][size]` `“50005”`

Data Values

Name	Size	Value
`opCode`	`Byte`	`5`

Response

Type	Response
	No response from server

5.S_QUIT

Input Message

Type	Data
String	[opCode][size] “60005”

Data Values

Name	Size	Value
opCode	Byte	6

Response

Type	Response
	No response from server - Server shutdown

Glossary

Conventions

- **Client** - Client application Implements the above Protocol
- **Status** - Failure or Successful operation.
- **Response** - String data sent from server to client application.
- Server Cannot Start without a given users.txt file argument.
- All responses are in String format.
- Default Port - 6423.
- Server can run with customized port - given as second argument.
- Maximum number of clients - 20.
- Maximum mail recipients - 20.
- Maximum Mails in Database - 32,000.
- Maximum Username and Password length - 100 characters each.
- Maximum Subject length - 100 characters.
- Maximum Content length - 2000 characters.

Notes and Endpoints use cases

- The mail_server.c and mail_client.c source code does not use external .c files and each implemented in one code file.
- The server implementation is based on server-database model were each mail of each user is saved in a global mail database with a specific id number. The mails database returns specific mail and response to query relative with id number.
- Each user object in the server holds a list of mails id`s that use for creating query`s to the mails database.
- All emails in the database **cannot** be erased. A deleted mail is marked as trashed and the user cannot access this email anymore.
- The client applications supports command correction mechanism. When the user entered the wrong command – the client application will print the expected command to the standard output.
- The server – as instructed – can be shut down by using the ***"CTRL-C"*** command in the server side.
- The client can remotely shut off the server by using ***"S_QUIT"*** command at the client side