P3 - Group Messaging System

Submitted By : Abdul Kadir Khimani, Ayush Singh, Nayan Khanna
Submitted for : IS F462, Network Programming.
IC : Dr. Hari Babu

Design Decisions :

We implement a group messaging system using message queues with support for creating, joining and listing groups, sending and receiving messages to and from private users and groups and auto delete messages.

Implementation :

Every client makes a message queue which is used to send messages to the server. The server also has a message queue through which it receives request messages from clients and sends back response messages.

Execution flow :

- ***Server***

    1. Server is continuously running and waiting for request messages from clients. Server handles these requests iteratively.
    2. The client message contains the necessary information to complete the request. The server receives the request message, reads the contents and processes it accordingly.
    3. It then creates a response message and sends it to the client.

    Group Management :
    The server keeps track of all the groups that exist, including all the users that are a part of the group and all the messages that were sent on the group. Group members are tracked with an array. When a new member joins the group, their name is added to the member list array. Messages are kept track similarly, when a user sends a message on the group, it is added to the message array.

    Auto Deletion Timer :
    When a user sends a message with an auto delete timeout, the timeout time is sent along with the message. The server stores the timeout time along with the message in a structure like this. The server also stores the UNIX timestamp at which this message was received.

```
typedef struct gmsg {
    char data[MAX_SIZE]; // MSG
    int t; // AUTO DELETE TIMEOUT
    unsigned long ct; // TIME OF CREATION OF MESSAGE
}gmsg;
```

The server continuously checks the current UNIX timestamp and subtracts it with the UNIX timestamp(ct) of all messages in all groups to find out if the timeout for any message has expired. If this is the case, the message is deleted.

- ***Client*** *:*

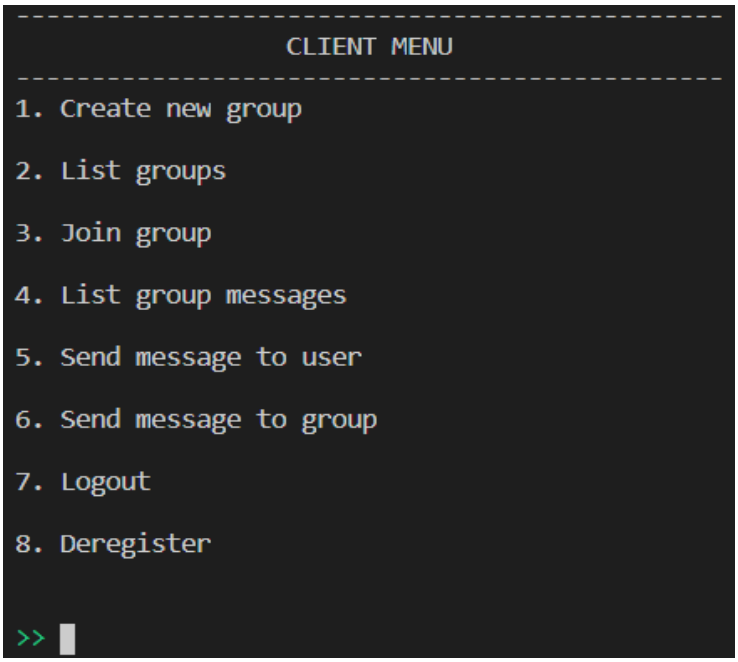
  Client enters a username in order to login. A new client is registered with the server automatically after it logs in for the first time.

  1. Client creates a child process and a parent process.
  2. Parent process shows the menu with options to the user and sends appropriate messages to the server for a particular option and waits for response.
  3. Child process loops forever reading response messages from the server.
  4. **A logged in client is online and a logged out client is offline**. On receiving the message, the child processes the response, i.e. creates a message history for each group, if it is needed.
  5. Client session includes all message history from login to logout. The group message history is persistent as long as the client is logged in(online mode) and the client can see all the messages it has received for a group during the current session by choosing the option from the menu.
  6. As soon as the client logs out, the message history is deleted. Message history is recorded using a shared memory as the child receives messages and we need the state to be the same across parent and child.
  7. When the client is logged out(offline mode), it can still receive messages from other clients which are delivered to the client once the client logs in(online mode), thus starting a new message history session.
  8. Deregistering deletes everything about a client from the system, meaning it is no longer a part of the messaging system. Its message queue is deleted, and the server also removes the client from the system.

Usage :
Change to the directory in which the source code is present, then
Compile :
In the bash terminal, issue command: **make**
Run :
In the bash terminal, issue command:
For server : **./server**
For client : **./client**

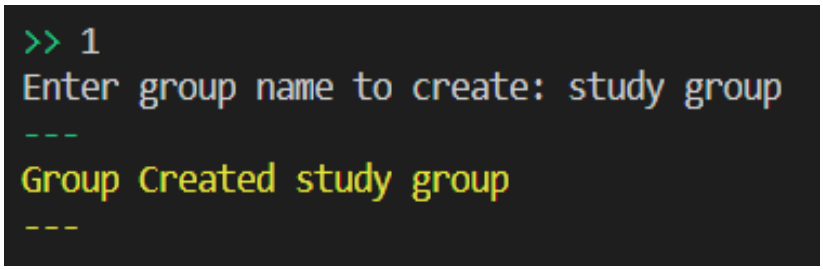
User Interface :

1. The client first prompts the user to enter their username in order to login into the application.
2. This opens up the client menu, which has all the various options the user can choose from.



3. The user can enter their choice and respectively the required functions will execute. After the function is completed, the user is returned back to the client menu.

- Creating a group :
  After the user enters '1' in the client menu, they are prompted to enter the name of the group to be created.



- Listing all groups :
  A new group 'study group 2' was created for this example.
  The user enters '2' in the client menu, they are displayed all the groups that exist on this server along with whether they are a member of the group or not.

```
>> 2
---
Group name : study group - Member
---
Group name : study group 2 - Member
---
```

- Joining a group :
  We create a new user for this example. Initially, they will not belong to any group.

```
>> 2
---
Group name : study group - Not Member
---
Group name : study group 2 - Not Member
---
```

After the user enters '2' in the client menu, they are prompted to enter the name of the group to join. For this example, we will join the 'study group' group created above.

When a new user joins a group, they receive all previously sent messages as well, if they join before the message timer for each message expires.

```
>> 3
Enter group name to join: study group
---
Added to the group study group
---
Messages:
first message - john
---
second message - bob
---
```

Now if we list the groups once more, we see the new user has successfully joined the 'study group' group.

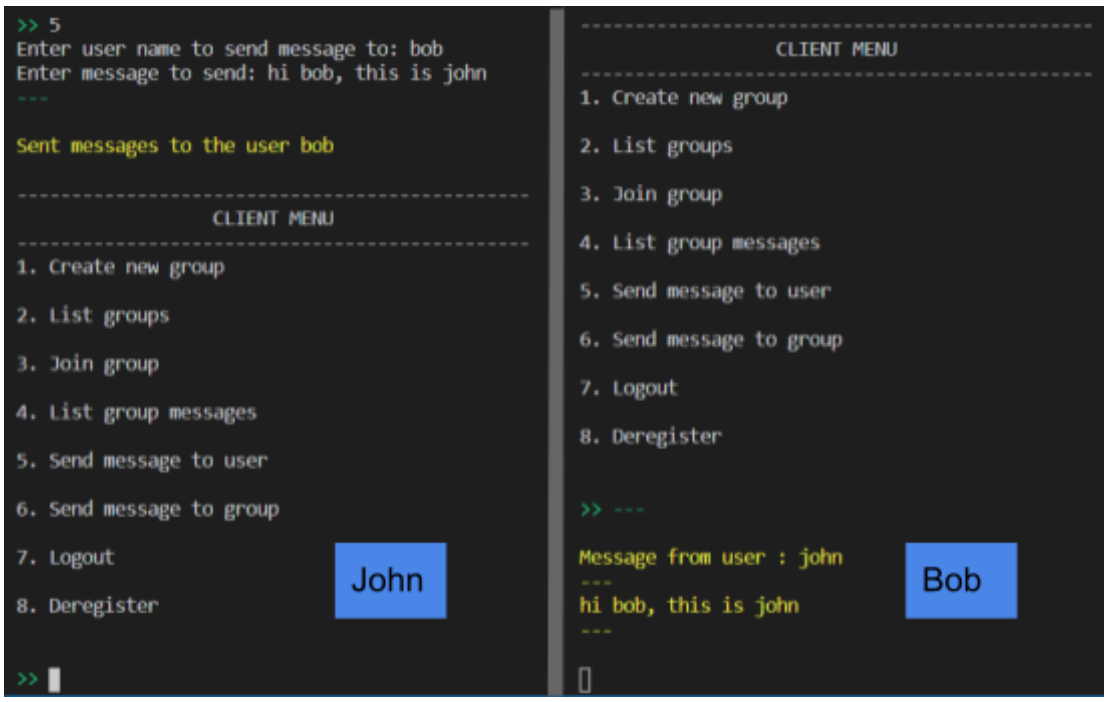
```
>> 2
---
Group name : study group - Member
---
Group name : study group 2 - Not Member
---
```

- List group messages :
  A few sample messages were sent for this example.
  On entering '4' in the client menu, the user is prompted to enter the name of the group to display messages from.

```
>> 4
Enter group name to display messages from: study group
Group Created study group
---
first message - john
---
second message - bob
---
```

The messages are only displayed for the current session. If the current session is ended by logging out, the message history is deleted. After that if a user logins, the pending messages are delivered, if any, creating a new message history.

- Sending a private message to a user :



The receiving client instantly receives the message.

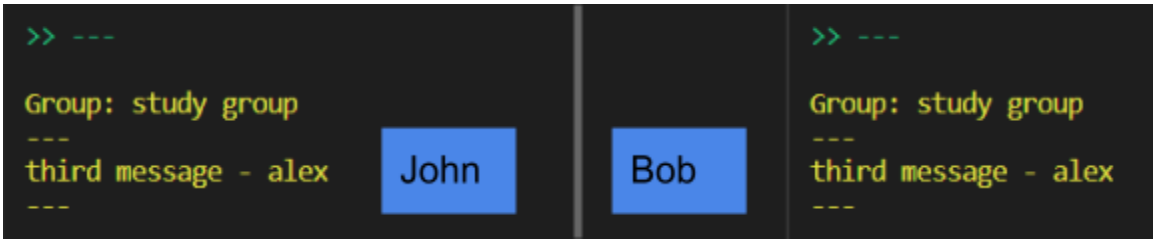
- Sending a message on a group :
  Another user alex was created for this example. His group message is received by both john and bob.

```
>> 6
Enter group name to send message to: study group
Enter message to send: third message
Auto delete timer duration(0 for no timer): 0
---

Group: study group
---
third message - alex
---

---

Sent messages to the group study group
---
```



Logging out
When a user logs out, the message queue associated with their username is not deleted. This way the user can continue to receive messages even when they're offline (logged out).

Deregistering
This option will delete the associated message queue of the user. User leaves the system forever.

END