Simple Message Box

DSA Hackathon

Name	SRN
Sujay R	PES1UG22AM168
Tejas N Torke	PES1UG22AM176
Yash Nagraj	PES1UG22AM197
Suki Perumal	PES1UG22AM180

This simple message box system was designed with a focus on using data structures that are efficient for the common operations one would expect to perform on such a system, namely appending messages, iterating over messages, and searching through messages.

Data Structure Usage and Justification:

For storing messages, a singly linked list was chosen. This data structure allows for efficient append operations, as new messages can be added to the end of the list without the need to resize an underlying array, which would be required if a dynamic array or similar structure were used. The linked list also ensures that we can iterate over the messages in a straightforward manner, which is necessary for displaying messages to users and for implementing user-friendly search functionality.

The linked list provides constant time insertion and deletion from the front, which may also be beneficial if future features require such operations. However, the downside is that it doesn't provide constant-time random access, but since our application does not require this feature frequently, the linked list is a justified choice.

Abstract Data Types (ADTs) Used:

The `Message` and `MessageBox` types serve as the ADTs in this system. The `Message` type encapsulates the content of a message along with a sender's username and a timestamp. The `MessageBox` type represents the overall collection of messages and abstracts the underlying linked list data structure from the user.

Features and Functionalities:

The message box provides several features:

- **Sending Messages:** Users can add new messages to the system. Each message includes the text entered by the user, a timestamp for when it was sent, and the username of the sender.
- **Receiving/Displaying Messages:** Users can view all messages sent in the system sorted by the order in which they were received.
- **Searching Messages:** Users can search through the message history using regular expressions. This powerful feature allows users to find messages that match complex patterns, rather than just simple string searches.
- **Filtering Messages by Username:** Users can filter the messages to only view the ones sent by a particular user, which can be useful when the message history grows large and conversations become interwoven.

Assumptions Made:

Several assumptions have been made throughout the design and implementation of the message box:

- Messages will be primarily appended at the end, and the history will not require frequent deletions. This assumption justified the choice of a linked list over other structures, such as balanced trees.
- The search operation is not the primary function used in the system; thus, sub-optimal search time complexity (linear) was acceptable in favor of easier implementation and understanding.
- Users interacting with the system have some knowledge of regular expressions for the search functionality to be fully utilized.
- The username provided by the user is accurate. No additional verification mechanism (such as user authentication) has been implemented in this simple system.
- Memory usage is sufficient to store all messages in a linked list structure. There is no need for persisting messages across system shutdowns, so database storage or serialization was not considered.

This message box system is an example of applying distinct data structures to address specific requirements of an application, balancing memory usage, system efficiency, and user-friendliness. The data structure choice ensures an extensible foundation upon which additional features and scalability improvements could be built.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <regex.h>

#define MESSAGE_LENGTH 256
#define USERNAME_LENGTH 32

typedef struct Message {
      char content[MESSAGE_LENGTH];
      char username[USERNAME_LENGTH];
      time_t timestamp;
      struct Message* next;
} Message;

typedef struct MessageBox {
      Message* head;
      size_t size;
} MessageBox;

// Function to create a message
Message* create_message(const char* username, const char* text) {
      Message* new_message = (Message*)malloc(sizeof(Message));
      if (new_message) {
            strncpy(new_message->username, username, USERNAME_LENGTH);
            new_message->username[USERNAME_LENGTH - 1] = '\0'; // Ensure
null-termination
            strncpy(new_message->content, text, MESSAGE_LENGTH);
            new_message->content[MESSAGE_LENGTH - 1] = '\0'; // Ensure
null-termination
            new_message->timestamp = time(NULL);
```

```
            new_message->next = NULL;
      }
      return new_message;
}

// Initialize the message box with no messages
void initialize_message_box(MessageBox* mbox) {
      mbox->head = NULL;
      mbox->size = 0;
}

// Function to add a message to the message box
void add_message(MessageBox* mbox, Message* message) {
      if (mbox->head == NULL) {
            mbox->head = message;
      } else {
            Message* temp = mbox->head;
            while (temp->next != NULL) {
                  temp = temp->next;
            }
            temp->next = message;
      }
      mbox->size++;
}

// Function to display all messages in the message box
void display_messages(const MessageBox* mbox) {
      Message* temp = mbox->head;
      while (temp != NULL) {
            printf("Username: %s\n", temp->username);
            printf("Time: %s", ctime(&(temp->timestamp)));
            printf("Message: %s\n\n", temp->content);
            temp = temp->next;
      }
}

// Function to free all messages in the message box
void free_message_box(MessageBox* mbox) {
      Message* temp = mbox->head;
      while (temp != NULL) {
            Message* next = temp->next;
            free(temp);
```

```
            temp = next;
      }
      mbox->head = NULL;
      mbox->size = 0;
}

// Function to prompt user and send a message
void send_message(MessageBox* mbox) {
      char input[MESSAGE_LENGTH];
      char username[USERNAME_LENGTH];
      printf("Enter your username: ");
      fgets(username, USERNAME_LENGTH, stdin);
      username[strcspn(username, "\n")] = 0; // Remove trailing newline
character

      printf("Enter your message (max 255 characters): ");
      fgets(input, MESSAGE_LENGTH, stdin);
      input[strcspn(input, "\n")] = 0; // Remove trailing newline character

      Message* message = create_message(username, input);
      add_message(mbox, message);
      printf("Message sent at %s\n", ctime(&(message->timestamp)));
}

// Function to search and display messages that match the regular expression
void search_messages(const MessageBox* mbox) {
      char pattern[MESSAGE_LENGTH];
      printf("Enter regex pattern to search for: ");
      fgets(pattern, MESSAGE_LENGTH, stdin);
      pattern[strcspn(pattern, "\n")] = 0; // Remove trailing newline
character

      regex_t regex;
      int reti;
      reti = regcomp(&regex, pattern, REG_EXTENDED);
      if (reti) {
            printf("Could not compile regex\n");
            return;
      }

      Message* temp = mbox->head;
      while (temp != NULL) {
```

```
            reti = regexec(&regex, temp->content, 0, NULL, 0);
            if (!reti) {
                  printf("Time: %sMessage: %s\n\n", ctime(&(temp->timestamp)),
temp->content);
            }
            temp = temp->next;
      }

      regfree(&regex);
}

// Function to
void filter_messages_by_username(const MessageBox* mbox, const char* username)
{
      Message* temp = mbox->head;
      while (temp != NULL) {
            if (strcmp(temp->username, username) == 0) {
                  printf("Username: %s\n", temp->username);
                  printf("Time: %s", ctime(&(temp->timestamp)));
                  printf("Message: %s\n\n", temp->content);
            }
            temp = temp->next;
      }
}

int main() {
      MessageBox mbox;
      initialize_message_box(&mbox);

      int running = 1;
      while (running) {
            printf("1. Send a message\n");
            printf("2. Display all messages\n");
            printf("3. Search messages by regex\n");
            printf("4. Filter messages by username\n");
            printf("0. Exit\n");
            printf("Choose an option: ");

            int choice;
            scanf("%d", &choice);
            // Clear the input buffer
            while (getchar() != '\n');
```

```
            switch (choice) {
                  case 1:
                        send_message(&mbox);
                  break;
                  case 2:
                        display_messages(&mbox);
                  break;
                  case 3:
                        search_messages(&mbox);
                  break;
                  case 4: {
                        char username[USERNAME_LENGTH];
                        printf("Enter username to filter by: ");
                        fgets(username, USERNAME_LENGTH, stdin);
                        username[strcspn(username, "\n")] = 0; // Remove
trailing newline character
                        filter_messages_by_username(&mbox, username);
                        break;
                  }
                  case 0:
                        running = 0;
                  break;
                  default:
                        printf("Invalid option. Please try again.\n");
            }
      }

      free_message_box(&mbox);

      return 0;
}
```

Sample output:

```
1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 1
```

```
Enter your username: yash
Enter your message (max 255 characters): ey bro
Message sent at Mon Nov 20 15:02:46 2023

1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 1
Enter your username: torke
Enter your message (max 255 characters): hello bro
Message sent at Mon Nov 20 15:02:55 2023

1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 2
Username: yash
Time: Mon Nov 20 15:02:46 2023
Message: ey bro

Username: torke
Time: Mon Nov 20 15:02:55 2023
Message: hello bro

1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 3
Enter regex pattern to search for: ^hello.*$
Time: Mon Nov 20 15:02:55 2023
Message: hello bro
```

```
1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 3
Enter regex pattern to search for: bro
Time: Mon Nov 20 15:02:46 2023
Message: ey bro

Time: Mon Nov 20 15:02:55 2023
Message: hello bro

1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 4
Enter username to filter by: torke
Username: torke
Time: Mon Nov 20 15:02:55 2023
Message: hello bro

1. Send a message
2. Display all messages
3. Search messages by regex
4. Filter messages by username
0. Exit
Choose an option: 4
Enter username to filter by: yash
Username: yash
Time: Mon Nov 20 15:02:46 2023
Message: ey bro

1. Send a message
2. Display all messages
3. Search messages by regex
```

```
4. Filter messages by username
0. Exit
Choose an option: 0
```