**ARUN M**

**Networking lab**

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8888
#define BUFFER_SIZE 1024

void send_response(int client_sock, const char *message) {
    send(client_sock, message, strlen(message), 0);
}

void handle_client(int client_sock) {
    char buffer[BUFFER_SIZE] = {0};

    send_response(client_sock, "220 Simple SMTP Server\r\n");

    while (1) {
        memset(buffer, 0, sizeof(buffer));
        ssize_t bytes_received = recv(client_sock, buffer, sizeof(buffer) - 1, 0);
        if (bytes_received <= 0) break;

        printf("Client: %s", buffer);

        if (strncmp(buffer, "HELO", 4) == 0) {
            send_response(client_sock, "250 Hello, pleased to meet you\r\n");
```

```
        }
        else if (strncmp(buffer, "MAIL FROM:", 10) == 0) {
            send_response(client_sock, "250 Sender OK\r\n");
        }
        else if (strncmp(buffer, "RCPT TO:", 8) == 0) {
            send_response(client_sock, "250 Recipient OK\r\n");
        }
        else if (strncmp(buffer, "DATA", 4) == 0) {
            send_response(client_sock, "354 End data with <CR><LF>.<CR><LF>\r\n");
        }
        else if (strcmp(buffer, ".\r\n") == 0) {
            send_response(client_sock, "250 Message accepted for delivery\r\n");
        }
        else if (strncmp(buffer, "QUIT", 4) == 0) {
            send_response(client_sock, "221 Bye\r\n");
            break;
        }
        else {
            send_response(client_sock, "500 Command not recognized\r\n");
        }
    }

    close(client_sock);
    printf("Client disconnected.\n");
}

int setup_server_socket() {
    int server_sock;
    struct sockaddr_in server_addr;

    // Create socket
```

```
    server_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (server_sock == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind failed");
        close(server_sock);
        exit(EXIT_FAILURE);
    }

    if (listen(server_sock, 5) < 0) {
        perror("Listen failed");
        close(server_sock);
        exit(EXIT_FAILURE);
    }

    printf("SMTP Server listening on port %d...\n", PORT);
    return server_sock;
}

int main() {
    int server_sock = setup_server_socket();
    int client_sock;
    struct sockaddr_in client_addr;
    socklen_t client_len = sizeof(client_addr);
```

```
    while (1) {
        client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &client_len);
        if (client_sock < 0) {
            perror("Accept failed");
            continue;
        }

        printf("Client connected...\n");
        handle_client(client_sock);
    }

    close(server_sock);
    return 0;
}
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define SERVER_IP "127.0.0.1"
#define PORT 8888


#define MAX_EMAIL     256
#define MAX_SUBJECT   512
#define MAX_MESSAGE   1024
#define MAX_COMMAND   512
```

```
#define MAX_RESPONSE  1024


void send_command(int sock, const char *command) {
    char response[MAX_RESPONSE] = {0};

    send(sock, command, strlen(command), 0);
    recv(sock, response, sizeof(response) - 1, 0);
    printf("Server: %s", response);
}

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char sender[MAX_EMAIL], recipient[MAX_EMAIL], subject[MAX_SUBJECT];
    char message[MAX_MESSAGE], temp[MAX_MESSAGE];


    printf("Enter sender email: ");
    fgets(sender, sizeof(sender), stdin);
    sender[strcspn(sender, "\n")] = 0;

    printf("Enter recipient email: ");
    fgets(recipient, sizeof(recipient), stdin);
    recipient[strcspn(recipient, "\n")] = 0;

    printf("Enter subject: ");
    fgets(subject, sizeof(subject), stdin);
    subject[strcspn(subject, "\n")] = 0;
```

```
printf("Enter message body (end with a single dot '.' on a new line):\n");
message[0] = '\0';
while (1) {
   fgets(temp, sizeof(temp), stdin);
   if (strcmp(temp, ".\n") == 0) break;
   if (strlen(message) + strlen(temp) >= MAX_MESSAGE - 1) {
      printf("Message too long. Truncating.\n");
      break;
   }
   strcat(message, temp);
}

sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1) {
   perror("Socket creation failed");
   exit(EXIT_FAILURE);
}

server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);


if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
   perror("Connection failed");
   close(sock);
   exit(EXIT_FAILURE);
}


char buffer[MAX_RESPONSE] = {0};
```

```
    recv(sock, buffer, sizeof(buffer) - 1, 0);
    printf("Server: %s", buffer);



    send_command(sock, "HELO localhost\r\n");

    char mail_from[MAX_COMMAND];
    snprintf(mail_from, sizeof(mail_from), "MAIL FROM:<%s>\r\n", sender);
    send_command(sock, mail_from);

    char rcpt_to[MAX_COMMAND];
    snprintf(rcpt_to, sizeof(rcpt_to), "RCPT TO:<%s>\r\n", recipient);
    send_command(sock, rcpt_to);

    send_command(sock, "DATA\r\n");

    char email_content[MAX_MESSAGE + MAX_SUBJECT + 100];
    snprintf(email_content, sizeof(email_content),
            "Subject: %s\r\n\r\n%s\r\n.\r\n", subject, message);
    send_command(sock, email_content);

    send_command(sock, "QUIT\r\n");

    close(sock);
    return 0;
}
```

```
Enter sender email: 1234@gmail
Enter recipient email: abc@gmail
Enter subject: smtp
Enter message body (end with a single dot '.' on a new line):
hello
.
Server: 220 Simple SMTP Server
Server: 250 Hello, pleased to meet you
Server: 250 Sender OK
Server: 250 Recipient OK
Server: 354 End data with <CR><LF>.<CR><LF>
Server: 500 Command not recognized
Server: 221 Bye
```

```
SMTP Server listening on port 8888...
Client connected...
Client: HELO localhost
Client: MAIL FROM:<1234@gmail>
Client: RCPT TO:<abc@gmail>
Client: DATA
Client: Subject: smtp

hello

.
Client: QUIT
Client disconnected.
```