IT Assignment 5

Manish Kumar

2020ITB0007

Using Own SMTP Server and Client

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <stdbool.h>

#define domain "server.smtp.com"

int main(int argc, char *argv[])
{
    int sockfd, newsockfd, n, portno, clilen;
    struct sockaddr_in serv_addr, cli_addr;
    char servers[2][100] = {"smtp.gmail.com", "smtp.yahoo.com"};
    char buff[10240]; // used for buffer the incoming data from client
    char command[50]; // used to read command RECEIVED from client

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        perror("\nError occured while creating the socket!\n");
    }

    bzero((char *)&serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);

    if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        perror("\nSocket binding failed!\n");
    }

    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
    if (newsockfd < 0)
    {
        perror("\nError occured while accepting on socket!\n");
    }

    // Implementing responses which SMTP Server sends to Client for Acknowledgement in the
process of Email Receiving
    do
    {
        bzero(buff, 10240);
        n = read(newsockfd, buff, 10239);
        if (n < 0)
        {
```

```
    printf("\nError occured while reading from socket!\n");
    break;
}
else
{
    buff[n] = '\0';
    if (strstr(buff, "HELO") != NULL)
    {
        printf("RECEIVED : %s", buff);
        bzero(buff, 10240);
        strcpy(buff, "250 Hello ");
        strcat(buff, domain);
        printf("SENT : %s\n\n", buff);
        n = write(newsockfd, buff, strlen(buff));
        if (n < 0)
        {
            perror("Error occured while writing to socket!");
        }
    }
    else if (strstr(buff, "MAIL FROM") != NULL)
    {
        printf("RECEIVED : %s", buff);
        bzero(buff, 10240);
        strcpy(buff, "250 OK");
        printf("SENT : %s\n\n", buff);
        n = write(newsockfd, buff, strlen(buff));
        if (n < 0)
        {
            perror("Error occured while writing to socket!");
        }
    }
```

```
else if (strstr(buff, "RCPT TO") != NULL)
{
    printf("RECEIVED : %s", buff);
    bzero(buff, 10240);
    strcpy(buff, "250 OK");
    printf("SENT : %s\n\n", buff);
    n = write(newsockfd, buff, strlen(buff));
    if (n < 0)
    {
        perror("Error occured while writing to socket!");
    }
}
else if (strstr(buff, "DATA") != NULL)
{
    printf("RECEIVED : %s", buff);
    bzero(buff, 10240);
    strcpy(buff, "354 Send message content; end with <CRLF>.<CRLF>");
    printf("SENT : %s\n\n", buff);
    n = write(newsockfd, buff, strlen(buff));
    if (n < 0)
    {
        perror("Error occured while writing to socket!");
    }

    bzero(buff, 10240);
    n = read(newsockfd, buff, 10239);
    if (n < 0)
    {
        printf("\nError occured while reading from socket!\n");
        break;
    }
```

```
            printf("\n\n----------| Received Email Header & Content | ---------\n\n%s\n", buff);
            printf("------------------------------------------------------------\n\n");


            bzero(buff, 10240);
            n = read(newsockfd, buff, 10239);
            if (n < 0)
            {
                printf("\nError occured while reading from socket!\n");
                break;
            }


            if (strstr(buff, ".") != NULL)
            {
                printf("RECEIVED : %s", buff);
                bzero(buff, 10240);


                strcpy(buff, "250 OK, message accepted for delivery.");
                printf("SENT : %s\n\n", buff);
                n = write(newsockfd, buff, strlen(buff));
                if (n < 0)
                {
                    perror("Error occured while writing to socket!");
                }
            }
        }


        else if (strstr(buff, "QUIT") != NULL)
        {
            break;
        }
    }
    } while (strcmp(buff, "QUIT") != 0);


    printf("RECEIVED : %s", buff);
    bzero(buff, 10240);
    strcpy(buff, "221 Bye");
    printf("SENT : %s\n\n", buff);
    n = write(newsockfd, buff, strlen(buff));
    if (n < 0)
    {
        perror("Error occured while writing to socket!");
    }
    printf("\nConnection closed successfully with the client!\n\n");


    return 0;
}
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>
#include <time.h>


#define domain "smtp-relay.sendinblue.com"
char mail_from[100];
char mail_to[100];
char *Mail_Header(char *from, char *to, char *sub, char *content)

{

   time_t t;

   time(&t);


   char *header = NULL;

   char date[26];

   char DATE_h[8 + strlen(date)];

   char sender[8 + strlen(from)]; // FROM: sender's_email\r\n

   char recep[6 + strlen(to)];   // TO: recepient's_email\r\n

   char subject[11 + strlen(sub)];

   char content_a[1 + strlen(content) + 2 + 1 + 1];

   strftime(date, (33), "%a %d %b %Y %H:%M:%S", localtime(&t));

   sprintf(DATE_h, "DATE: %s\r\n", date);

   sprintf(sender, "FROM: %s\r\n", from);

   sprintf(subject, "Subject: %s\r\n", sub);

   sprintf(recep, "TO: %s\r\n", to);

   // extra \n is used to end the header part


   sprintf(content_a, "%s\r\n", content);


   int header_length = strlen(DATE_h) + strlen(sender) + strlen(subject) + strlen(recep) +
strlen(content_a);


   header = (char *)malloc(header_length * sizeof(char));


   memcpy(&header[0], &DATE_h, strlen(DATE_h));

   memcpy(&header[0 + strlen(DATE_h)], &sender, strlen(sender));

   memcpy(&header[0 + strlen(DATE_h) + strlen(sender)], &subject, strlen(subject));

   memcpy(&header[0 + strlen(DATE_h) + strlen(sender) + strlen(subject)], &recep, strlen(recep));

   memcpy(&header[0 + strlen(DATE_h) + strlen(sender) + strlen(subject) + strlen(recep)],
&content_a, strlen(content_a));
```

```
    return header;
}

int main(int argc, char *argv[])
{
    int socket_id, n;
    int portno;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char cname[256];
    char buff[10240];

    if (argc < 3)
    {
        perror("\nPlease enter the hostname and port number.\n");
        exit(0);
    }
    portno = atoi(argv[2]);

    socket_id = socket(AF_INET, SOCK_STREAM, 0);

    if (socket_id < 0)
    {
        perror("\nError occured while opening the socket!\n");
        exit(0);
    }
    server = gethostbyname(argv[1]);

    if (server == NULL)
    {
```

```
    perror("\nError: No such host found!\n");
    exit(0);
}

bzero((char *)&serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
serv_addr.sin_port = htons(portno);

// connect to the server
int connect_id;
connect_id = connect(socket_id, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
if (connect_id < 0)
{
    perror("Error occured while connecting to server...\n");
}

// Implementing commands which are used b/w Client and SMTP Server for communication
do
{
    printf("Enter the command : ");
intake:
    gets(cname);
    // cname[strlen(cname) + 1] = '\0';

    char code[4]; // to store the 3 digit response code received from server

    if (strcasecmp(cname, "HELO") == 0)
    {
        bzero(buff, 10240);
        strcpy(buff, "HELO ");
```

```
strcat(buff, domain);
strcat(buff, "\r\n");
n = write(socket_id, buff, strlen(buff));
if (n < 0)
{
   printf("\nError occured while writing to socket!\n");
}
printf("\nCLIENT : %s", buff); // HELO domain
bzero(buff, 10240);
n = read(socket_id, buff, 10239);
if (n < 0)
{
   printf("\nError occured while reading from socket!\n");
}
printf("SERVER : %s\n", buff); // 250 Hello domain

// checking error
code[0] = buff[0];
code[1] = buff[1];
code[2] = buff[2];
code[3] = '\0';

if (strcmp(code, "250") == 0)
{
   printf("\nGo to next command...\n\n");
}
else
{
   printf("\nError occured!\n\n");
}
flush(stdin);
```

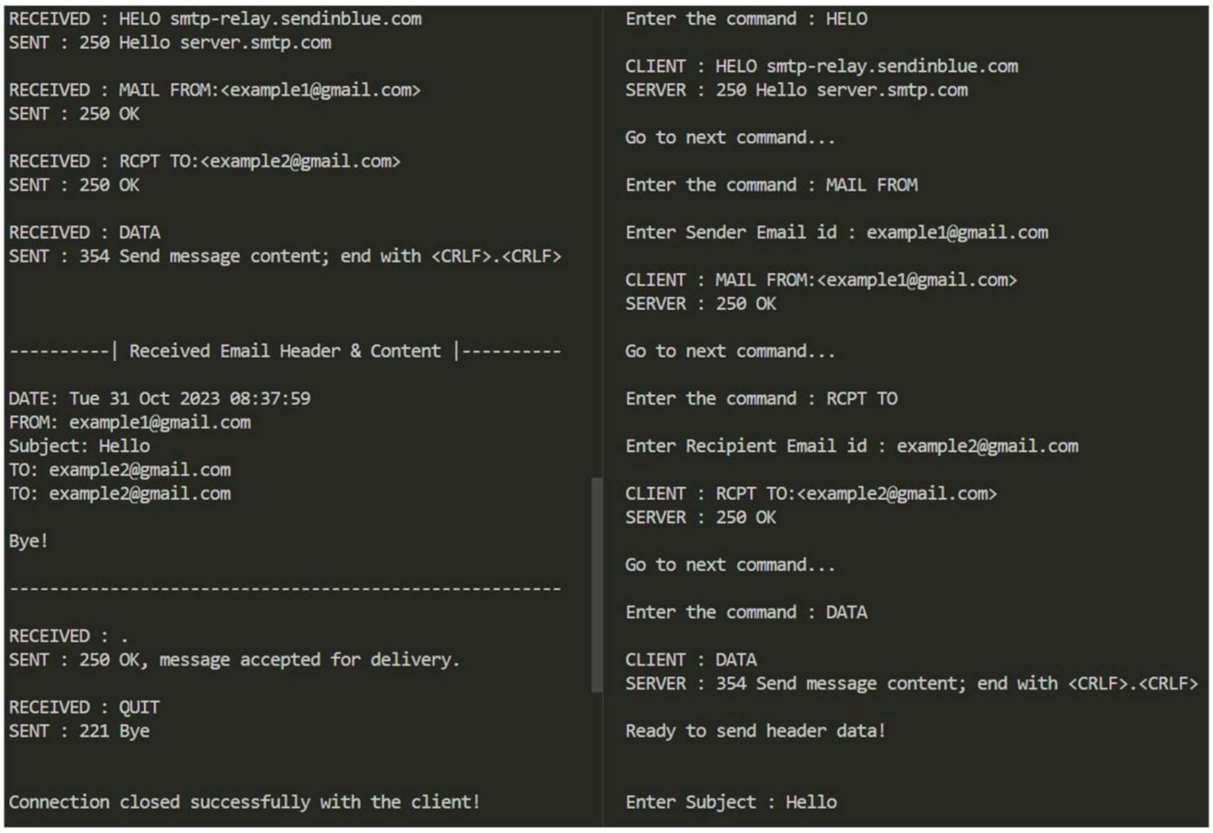
```
}
else if (strcasecmp(cname, "MAIL FROM") == 0)
{
    bzero(buff, 10240);
    printf("\nEnter Sender Email id : ");
    scanf("%s", mail_from);
    strcpy(buff, "MAIL FROM:<");
    strcat(buff, mail_from);
    strcat(buff, ">");
    strcat(buff, "\r\n");
    n = write(socket_id, buff, strlen(buff));
    if (n < 0)
    {
        printf("\nError occured while writing to socket!\n");
    }
    printf("\nCLIENT : %s", buff); // MAIL FROM:<your email id>
    bzero(buff, 10240);
    n = read(socket_id, buff, 10239);
    if (n < 0)
    {
        printf("\nError occured while reading from socket!\n");
    }
    printf("SERVER : %s\n", buff); // 250 OK

    // checking error
    code[0] = buff[0];
    code[1] = buff[1];
    code[2] = buff[2];
    code[3] = '\0';

    if (strcmp(code, "250") == 0)
    {
        printf("\nGo to next command...\n\n");
    }
    else
    {
        printf("\nError occured!\n\n");
    }
    flush(stdin);
}
else if (strcasecmp(cname, "RCPT TO") == 0)
{
    bzero(buff, 10240);
    printf("\nEnter Recipient Email id : ");
    scanf("%s", mail_to);
    strcpy(buff, "RCPT TO:<");
    strcat(buff, mail_to);
    strcat(buff, ">");
    strcat(buff, "\r\n");
    n = write(socket_id, buff, strlen(buff));
    if (n < 0)
    {
        printf("\nError occured while writing to socket!\n");
    }
    printf("\nCLIENT : %s", buff); // RCPT TO:<your email id>
    bzero(buff, 10240);
    n = read(socket_id, buff, 10239);
    if (n < 0)
    {
        printf("\nError occured while reading from socket!\n");
    }
    printf("SERVER : %s\n", buff); // 250 OK
    // checking error
    code[0] = buff[0];
    code[1] = buff[1];
    code[2] = buff[2];
    code[3] = '\0';

    if (strcmp(code, "250") == 0)
    {
        printf("\nGo to next command...\n\n");
    }
    else
    {
        printf("\nError occured!\n\n");
    }
    flush(stdin);
}
else if (strcasecmp(cname, "DATA") == 0)
{
    bzero(buff, 10240);
    strcpy(buff, "DATA");
    strcat(buff, "\r\n");
    n = write(socket_id, buff, strlen(buff));
    if (n < 0)
    {
        printf("\nError occured while writing to socket!\n");
    }
    printf("\nCLIENT : %s", buff); // DATA
    bzero(buff, 10240);
    n = read(socket_id, buff, 10239);
    if (n < 0)
```

```
{
    printf("\nError occured while reading from socket!\n");
}
printf("SERVER : %s\n", buff); // 354 Send message content; end with <CRLF>.<CRLF>

// checking error
code[0] = buff[0];
code[1] = buff[1];
code[2] = buff[2];
code[3] = '\0';

if (strcmp(code, "354") == 0)
{
    printf("\nReady to send header data!\n\n");
}
else
{
    printf("\nError occured!\n\n");
}

// creating a mail header
char sub[150];
char content[450];
printf("\nEnter Subject : ");
scanf("%[^\n]", sub);

printf("\nEnter content : (Press Tab and Enter Key to end)\n");
scanf("%[^\t]", content);

bzero(buff, 10240);
// Mail_header function declared above
```

```
strcpy(buff, Mail_Header(mail_from, mail_to, sub, content)); // assigning header to buffer


n = write(socket_id, buff, strlen(buff));
if (n < 0)
{
    printf("\nError occured while writing to socket!\n");
}
printf("\nCLIENT : ====| Mail header & content |====\n%s\n", buff); // header content


bzero(buff, 10240);
strcpy(buff, ".\r\n");
n = write(socket_id, buff, strlen(buff));
if (n < 0)
{
    printf("\nError occured while writing to socket!\n");
}


bzero(buff, 10240);
n = read(socket_id, buff, 10239);
if (n < 0)
{
    printf("\nError occured while reading from socket!\n");
}
printf("SERVER : %s\n", buff); // 250 OK


// checking error
code[0] = buff[0];
code[1] = buff[1];
code[2] = buff[2];
code[3] = '\0';
```

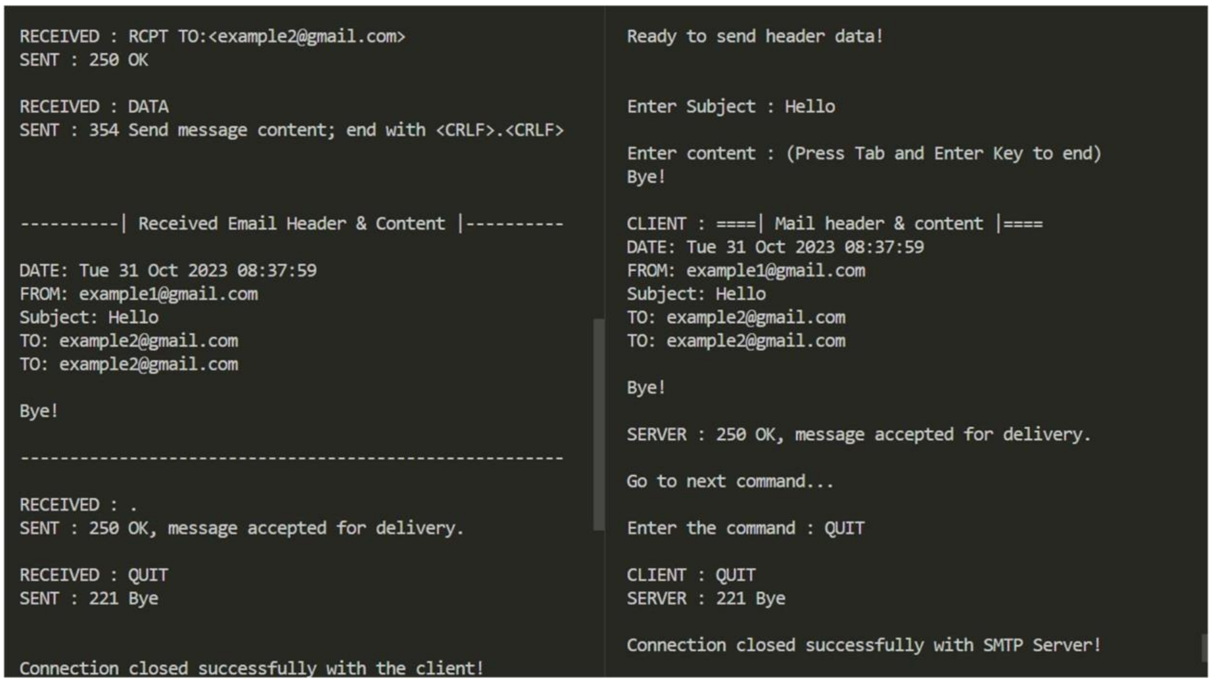
```
    if (strcmp(code, "250") == 0)
    {
        printf("\nGo to next command...\n\n");
    }
    else
    {
        printf("\nError occured!\n\n");
    }
    flush(stdin);
}

else if (strcasecmp(cname, "QUIT") == 0)
{
    bzero(buff, 10240);
    strcpy(buff, "QUIT");
    strcat(buff, "\r\n");
    n = write(socket_id, buff, strlen(buff));
    if (n < 0)
    {
        printf("\nError occured while writing to socket!\n");
    }
    printf("\nCLIENT : %s", buff); // QUIT
    bzero(buff, 10240);
    n = read(socket_id, buff, 10239);
    if (n < 0)
    {
        printf("\nError occured while reading from socket!\n");
    }
    printf("SERVER : %s\n", buff); // 221 Bye

    // checking error
```

```
            code[0] = buff[0];

            code[1] = buff[1];

            code[2] = buff[2];

            code[3] = '\0';


            if (strcmp(code, "221") == 0)

            {

                printf("\nConnection closed successfully with SMTP Server!\n\n");

            }

            else

            {

                printf("\nError occured!\n\n");

            }

            flush(stdin);

        }

        else

        {

            strcpy(cname, "");

            goto intake;

        }


    } while (strcmp(cname, "QUIT") != 0);

}
```



```
RECEIVED : HELO smtp-relay.sendinblue.com
SENT : 250 Hello server.smtp.com

RECEIVED : MAIL FROM:<example1@gmail.com>
SENT : 250 OK

RECEIVED : RCPT TO:<example2@gmail.com>
SENT : 250 OK

RECEIVED : DATA
SENT : 354 Send message content; end with <CRLF>.<CRLF>


----------| Received Email Header & Content |----------

DATE: Tue 31 Oct 2023 08:37:59
FROM: example1@gmail.com
Subject: Hello
TO: example2@gmail.com
TO: example2@gmail.com

Bye!

-------------------------------------------------------

RECEIVED : .
SENT : 250 OK, message accepted for delivery.

RECEIVED : QUIT
SENT : 221 Bye


Connection closed successfully with the client!
```

```
Enter the command : HELO

CLIENT : HELO smtp-relay.sendinblue.com
SERVER : 250 Hello server.smtp.com

Go to next command...

Enter the command : MAIL FROM

Enter Sender Email id : example1@gmail.com

CLIENT : MAIL FROM:<example1@gmail.com>
SERVER : 250 OK

Go to next command...

Enter the command : RCPT TO

Enter Recipient Email id : example2@gmail.com

CLIENT : RCPT TO:<example2@gmail.com>
SERVER : 250 OK

Go to next command...

Enter the command : DATA

CLIENT : DATA
SERVER : 354 Send message content; end with <CRLF>.<CRLF>

Ready to send header data!


Enter Subject : Hello
```



```
RECEIVED : RCPT TO:<example2@gmail.com>
SENT : 250 OK

RECEIVED : DATA
SENT : 354 Send message content; end with <CRLF>.<CRLF>


----------| Received Email Header & Content |----------

DATE: Tue 31 Oct 2023 08:37:59
FROM: example1@gmail.com
Subject: Hello
TO: example2@gmail.com
TO: example2@gmail.com

Bye!

-------------------------------------------------------

RECEIVED : .
SENT : 250 OK, message accepted for delivery.

RECEIVED : QUIT
SENT : 221 Bye


Connection closed successfully with the client!
```

```
Ready to send header data!


Enter Subject : Hello

Enter content : (Press Tab and Enter Key to end)
Bye!

CLIENT : ====| Mail header & content |====
DATE: Tue 31 Oct 2023 08:37:59
FROM: example1@gmail.com
Subject: Hello
TO: example2@gmail.com
TO: example2@gmail.com

Bye!

SERVER : 250 OK, message accepted for delivery.

Go to next command...

Enter the command : QUIT

CLIENT : QUIT
SERVER : 221 Bye

Connection closed successfully with SMTP Server!
```

Using SMTP Gmail

Steps to run code

```
sudo apt-get install libssl-dev

g++ code.c -L/usr/local/ssl/lib -lssl -lcrypto && ./a.out
```

Code

```
#include <sys/socket.h>

#include <sys/errno.h>

#include <netinet/in.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <resolv.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <openssl/bio.h>

#include <openssl/err.h>

#include <openssl/ssl.h>


#define BUFLEN 4096

#define SERVER "smtp.gmail.com"

#define PORT 587

#define EMAIL_LOGIN

"2020itb007.manish@students.iiests.ac.in" #define

EMAIL_PASSWORD "pmirpjypdkeikfdq"

#define SENDER_EMAIL

"2020itb007.manish@students.iiests.ac.in" #define

RECEIVER_EMAIL "manishmandal9734@gmail.com"

char *base64_encode(const char *data,

            size_t input_length,

            size_t *output_length)

{


  BIO *bio, *b64;
```

```
    BUF_MEM *bufferPtr;

    b64 = BIO_new(BIO_f_base64());
    bio = BIO_new(BIO_s_mem());
    bio = BIO_push(b64, bio);

    BIO_set_flags(bio, BIO_FLAGS_BASE64_NO_NL);
    BIO_write(bio, data, input_length);
    BIO_flush(bio);
    BIO_get_mem_ptr(bio, &bufferPtr);
    BIO_set_close(bio, BIO_NOCLOSE);
    BIO_free_all(bio);
    *output_length = (*bufferPtr).length;
    return (*bufferPtr).data;
}

int main()
{
    int sock;
    char *host, *cmd, *ip;
    char *enc_cmd;
    char command[128];
    char recvbuf[BUFLEN];
    int iResult;
    size_t out_len;

    struct hostent *hent;
    struct sockaddr_in sin;

    host = SERVER;
```

```
printf("Attempting to connect to %s...\n", host);
//
hent = gethostbyname(host);
if (hent == NULL)
{
   printf("gethostbyname failed: %d\n", errno);
   return -1;
}
printf("gethostbyname succeeded!\n");
ip = inet_ntoa(*(struct in_addr *)hent->h_addr_list[0]);
printf("Host IP: %s\n", ip);
//
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1)
{
   printf("socket failed: %d\n", errno);
   return -1;
}
printf("socket succeeded!\n");
//
bzero(&sin, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons(PORT);
sin.sin_addr.s_addr = inet_addr(ip);
//
iResult = connect(sock, (struct sockaddr *)&sin, sizeof(sin));
if (iResult < 0)
{
   printf("1 connect failed: %d\n", errno);
   return -1;
}
```

```
printf("connect succeeded\n");
//
bzero(recvbuf, BUFLEN);
iResult = recv(sock, recvbuf, BUFLEN - 1, 0);
if (iResult <= 0)
{
   printf("1 recv failed: %d\n", errno);
   return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
// cmd = "HELO smtp.gmail.com\r\n";
sprintf(command, "HELO %s\r\n", SERVER);

iResult = send(sock, command, strlen(command), 0);
if (iResult <= 0)
{
   printf("2 send failed: %d\n", errno);
   return -1;
}
printf("Byte(s) sent: %d\n", iResult);
//
bzero(recvbuf, BUFLEN);
iResult = recv(sock, recvbuf, BUFLEN - 1, 0);
if (iResult <= 0)
{
   printf("2 recv failed: %d\n", errno);
   return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
cmd = "STARTTLS\r\n";
iResult = send(sock, cmd, strlen(cmd), 0);
if (iResult <= 0)
{
   printf("3 send failed: %d\n", errno);
   return -1;
}
printf("Byte(s) sent: %d\n", iResult);
//
bzero(recvbuf, BUFLEN);
iResult = recv(sock, recvbuf, BUFLEN - 1, 0);
if (iResult <= 0)
{
   printf(" recv failed: %d\n", errno);
   return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
// OpenSLL region
OpenSSL_add_all_algorithms();
ERR_load_BIO_strings();
ERR_load_crypto_strings();
SSL_load_error_strings();
if (SSL_library_init() < 0)
{
   printf("Could not initialise SSL library!\n");
   ERR_print_errors_fp(stderr);
   return -1;
```

```
}
printf("SSL Library initialised\n");
SSL_CTX *ctx = SSL_CTX_new(SSLv23_client_method());
if (ctx == NULL)
{
   printf("ctx failed: %d\n", errno);
   ERR_print_errors_fp(stderr);
   return -1;
}
printf("ctx done!\n");
SSL *ssl = SSL_new(ctx);
if (ssl == NULL)
{
   printf("ssl failed: %d\n", errno);
   ERR_print_errors_fp(stderr);
   return -1;
}
printf("ssl done!\n");
SSL_set_fd(ssl, sock);
iResult = SSL_connect(ssl);
if (iResult < 0)
{
   printf("SSL connect failed!\n");
   ERR_print_errors_fp(stderr);
   return -1;
}
printf("SSL connect succeeded!\n");
sprintf(command, "HELO %s\r\n", SERVER);
iResult = SSL_write(ssl, command, strlen(command));
if (iResult <= 0)
{
    printf("SSL_write failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
//
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
cmd = "AUTH LOGIN\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write faile\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);

sprintf(command, "%s\r\n", EMAIL_LOGIN);
enc_cmd = base64_encode(command, strlen(command), &out_len);
iResult = SSL_write(ssl, enc_cmd, out_len);
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
cmd = "\r\n";
SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
sprintf(command, "%s\r\n", EMAIL_PASSWORD);
enc_cmd = base64_encode(command, strlen(command), &out_len);
iResult = SSL_write(ssl, enc_cmd, out_len);
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
cmd = "\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
```

```
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
sprintf(command, "MAIL FROM: <%s>\r\n", SENDER_EMAIL);
iResult = SSL_write(ssl, command, strlen(command));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) sent: %d\n", iResult);
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);

sprintf(command, "RCPT TO: <%s>\r\n", RECEIVER_EMAIL);
iResult = SSL_write(ssl, command, strlen(command));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
```

```
}
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
cmd = "DATA\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
```

```
cmd = "MIME-Version: 1.0\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
cmd = "Subject: SMTP Test Mail\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
//
cmd = "Content-Type:multipart/alternative;boundary=\"00000000000040d44d0608fa090a\"\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}

//
cmd = "\n--00000000000040d44d0608fa090a\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
//
cmd = "Content-Type: text/plain; charset=\"UTF-8\"\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
//
cmd = "Hello and welcome to Internet Technology!\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
//

cmd = "\n--00000000000040d44d0608fa090a\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
```

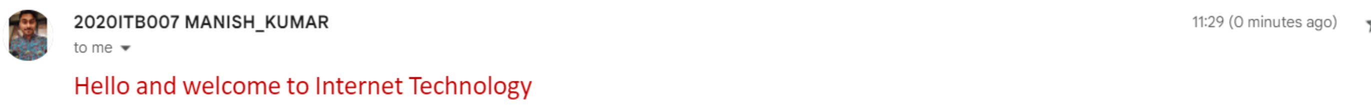
```
}
//
cmd = "Content-Type: text/html; charset=\"UTF-8\"\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
cmd = "<h1>Hello and welcome to Internet Technology</h1>\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
cmd = "\r\n.\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
cmd = "QUIT\r\n";
iResult = SSL_write(ssl, cmd, strlen(cmd));
if (iResult <= 0)
{
    printf("SSL_write failed\n");
    ERR_print_errors_fp(stderr);
    return -1;
}
bzero(recvbuf, BUFLEN);
iResult = SSL_read(ssl, recvbuf, BUFLEN - 1);
if (iResult <= 0)
{
    printf("SSL_read failed: %d\n", errno);
    ERR_print_errors_fp(stderr);
    return -1;
}
printf("Byte(s) received: %d\n", iResult);
printf("%s\n", recvbuf);
//
SSL_CTX_free(ctx);
printf("SSL closed!\n");
iResult = SSL_shutdown(ssl);
if (iResult == 0)
{
    printf("SSL shutdown in progress...\n");
    }
    iResult = SSL_shutdown(ssl);
    if (iResult == 1)
    {
        printf("SSL shutdown succeeded\n");
    }
    if (iResult == -1)
    {
        printf("SSL shutdown failed!\n");
    }
    //
    iResult = shutdown(sock, SHUT_RDWR);
    if (iResult == -1)
    {
        printf("shutdown failed: %d\n", errno);
        // return -1;
    }
    printf("shutdown succeeded\n");
    iResult = close(sock);
    if (iResult < 0)
    {
        printf("Error occurred while closing socket\n");
        return -1;
    }
    //
    return 0;
}
```

Output

```
Attempting to connect to smtp.gmail.com...
gethostbyname succeeded!
```

```
Host IP: 74.125.24.108
socket succeeded!
connect succeeded
Byte(s) received: 86
220 smtp.gmail.com ESMTP z19-20020a170902ee1300b001c726147a46sm192167plb.234 - gsmtp

Byte(s) sent: 21
Byte(s) received: 36
250 smtp.gmail.com at your service

Byte(s) sent: 10
Byte(s) received: 30
220 2.0.0 Ready to start TLS

SSL Library initialised
ctx done!
ssl done!
SSL connect succeeded!
Byte(s) sent: 21
Byte(s) received: 36
250 smtp.gmail.com at your service

Byte(s) sent: 12
Byte(s) received: 18
334 VXNlcm5hbWU6

Byte(s) sent: 56
Byte(s) sent: 56
Byte(s) received: 18
334 UGFzc3dvcmQ6
```

```
Byte(s) sent: 24

Byte(s) sent: 2

Byte(s) received: 20

235 2.7.0 Accepted


Byte(s) sent: 53

Byte(s) received: 74

250 2.1.0 OK z19-20020a170902ee1300b001c726147a46sm192167plb.234 - gsmtp


Byte(s) received: 74

250 2.1.5 OK z19-20020a170902ee1300b001c726147a46sm192167plb.234 - gsmtp


Byte(s) received: 75

354  Go ahead z19-20020a170902ee1300b001c726147a46sm192167plb.234 - gsmtp


Byte(s) received: 86

250 2.0.0 OK  1698720842 z19-20020a170902ee1300b001c72614726147a46sm192167plb.234 -
gsmtp


Byte(s) received: 90

221 2.0.0 closing connection z19-20020a170902ee1300b001c726147a46sm192167plb.234 - gsmtp


SSL closed!

SSL shutdown in progress...

shutdown succeeded
```



2020ITB007 MANISH_KUMAR
to me
11:29 (0 minutes ago)

Hello and welcome to Internet Technology