哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2023 春季

课程名称：计算机网络

实验名称：邮件客户端的设计与实现

学生班级：6 班

学生学号：200110625

学生姓名：柯炽炜

评阅教师：

报告成绩：

实验与创新实践教育中心制

2023 年 3 月

一、 实验详细设计

（注意不要完全照搬实验指导书上的内容，请根据你自己的设计方案来填写图文并茂地描述实验实现的所有功能和详细的设计方案及实验过程中的特色部分。）

1. 邮件发送客户端详细设计

1. 定义常量和变量：定义了一些常量和全局变量，包括缓冲区的大小和用于存储邮件内容的缓冲区。

```
#define MAX_SIZE 4095

char buf[MAX_SIZE+1];
```

2. `send_message` 函数：用于发送消息给服务器。它接受套接字文件描述符、缓冲区、大小、标志和消息作为参数。它会将消息打印到控制台并使用 `send` 函数发送给服务器。

```
void send_message(int s_fd, const void *buf, size_t size, int flags, const char *message) {
    printf(">>> %s \n", message);
    send(s_fd, buf, size, flags);
}
```

3. `receive_message` 函数：用于接收服务器的消息。它接受套接字文件描述符、缓冲区、大小、标志和错误消息作为参数。如果接收消息时发生错误，它会打印错误消息并退出程序。它还会在接收完消息后将缓冲区的末尾置为字符串结束符，并打印接收到的消息内容。

```
int receive_message(int s_fd, void *buf, size_t size, int flags, const char *error_message)  {
    int r_size;
    if ((r_size = recv(s_fd, buf, MAX_SIZE, 0)) == -1) {
        perror(error_message);
        exit(EXIT_FAILURE);
    }
    ((char *)buf)[r_size] = '\0';
    printf("%s", (char *)buf);
    return r_size;
}
```

4. `get_file_name` 函数：从给定的路径中解析出文件名。它接受一个路径作为参数，并使用 `strrchr` 函数查找最后一个斜杠字符，然后返回文件名的指针。

```
const char *get_file_name(const char *path) {
    // 从路径中解析文件名
    const char *p = strrchr(path, '/');
```

```
    if (p == NULL) {
        p = path;
    } else {
        p++;
    }
    return p;
}
```

5. `read_file` 函数：读取文件的内容并返回一个动态分配的缓冲区。它接受一个文件路径作为参数，使用标准库函数打开文件，获取文件大小，然后分配足够大的缓冲区来存储文件内容。最后，它将文件内容读取到缓冲区中，并在末尾添加字符串结束符。

```
char *read_file(const char *path) {
    FILE *fp = fopen(path, "r");
    if (fp == NULL) {
        return NULL;
    }
    // 获取文件大小
    fseek(fp, 0L, SEEK_END);
    long int size = ftell(fp);
    rewind(fp);
    // 获取文件内容
    char *data = NULL;
    if ((data = (char *)malloc(size + 1)) == NULL) {
        fclose(fp);
        return NULL;
    }
    // 写入结束符
    fread(data, size, 1, fp);
    data[size] = '\0';
    fclose(fp);
    return data;
}
```

6. `send_mail` 函数：发送邮件的主要函数。它接受收件人地址、邮件主题、邮件内容或包含邮件内容的文件路径以及附件路径作为参数。在函数内部，它使用指定的邮件服务器域名和端口号建立 TCP 连接。然后，它按照 SMTP 协议的规范，发送各种命令和数据给邮件服务器，完成邮件发送的过程。

1. 在函数的开头，定义了一些常量和变量。其中包括邮件服务器的主机名、端口号、认证所需的用户名和密码，以及其他辅助变量。

```
const char* end_msg = "\r\n.\r\n";
const char* host_name = "smtp.qq.com"; // TODO: Specify the mail server domain name
const unsigned short port = 25; // SMTP server port
```

```
const char* user = "838818923@qq.com"; // TODO: Specify the user
const char* pass = "tatboirwvsvsbegi"; // TODO: Specify the password
const char* from = "838818923@qq.com"; // TODO: Specify the mail address of the sender
```

2. 通过 `gethostbyname` 函数，根据邮件服务器的主机名获取其对应的 IP 地址。

```
char dest_ip[16]; // Mail server IP address
int s_fd; // socket file descriptor
struct hostent *host;
struct in_addr **addr_list;
int i = 0;
int r_size;
// Get IP from domain name
if ((host = gethostbyname(host_name)) == NULL)
{
    herror("gethostbyname");
    exit(EXIT_FAILURE);
}
addr_list = (struct in_addr **) host->h_addr_list;
while (addr_list[i] != NULL)
    ++i;
strcpy(dest_ip, inet_ntoa(*addr_list[i-1]));
```

3. 创建一个套接字（socket），并通过 `connect` 函数建立与邮件服务器的 TCP 连接。

```
if ((s_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket wrong");
    exit(EXIT_FAILURE);
}
struct sockaddr_in socket_address;
// 写入 protocol，端口号和 IP 地址
socket_address.sin_family = AF_INET;
socket_address.sin_port = htons(port);
socket_address.sin_addr.s_addr = inet_addr(dest_ip);
bzero(&(socket_address.sin_zero), 8);
connect(s_fd, (struct sockaddr*) &socket_address, sizeof(struct sockaddr));
```

4. 使用 `receive_message` 函数接收邮件服务器发送的欢迎消息，并打印在控制台上。

```
receive_message(s_fd, buf, MAX_SIZE, 0, "recv welcome");
```

5. 使用 `send_message` 函数发送 EHLO 命令给邮件服务器，并打印发送的命令，并且调用 `receive_message` 函数接收邮件服务器对 EHLO 命令的响应，并打印在控制台上。

```
// Send EHLO command and print server response
const char* EHLO = "EHLO kjm\r\n"; // TODO: Enter EHLO command here
send_message(s_fd, EHLO, strlen(EHLO), 0, "send EHLO");
// TODO: Print server response to EHLO command
receive_message(s_fd, buf, MAX_SIZE, 0, "recv EHLO");
```

6. 使用 `send_message` 函数发送认证请求命令 AUTH LOGIN，并打印发送的命令。并且使用 `receive_message` 函数接收邮件服务器对 AUTH LOGIN 命令的响应，并打印在控制台上。

```
// TODO: Authentication. Server response should be printed out.
// 认证请求
const char *AUTH_LOGIN = "AUTH LOGIN\r\n";
send_message(s_fd, AUTH_LOGIN, strlen(AUTH_LOGIN), 0, "send AUTH LOGIN");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv AUTH_LOGIN");
```

7. 对用户名和密码进行 Base64 编码，并发送给邮件服务器进行认证。

```
// 发送用户名
char *user_base64 = encode_str(user);
strcat(user_base64, "\r\n");
send_message(s_fd, user_base64, strlen(user_base64), 0, "send username");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv user");
free(user_base64);
// 发送密码
char *pass_base64 = encode_str(pass);
strcat(pass_base64, "\r\n");
send_message(s_fd, pass_base64, strlen(pass_base64), 0, "send password");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv pass");
free(pass_base64);
```

8. 使用 `send_message` 函数发送 MAIL FROM 命令，并打印发送的命令，并且使用 `receive_message` 函数接收邮件服务器对 MAIL FROM 命令的响应，并打印在控制台上。

```
// TODO: Send MAIL FROM command and print server response
sprintf(buf, "MAIL FROM:<%s>\r\n", user);
send_message(s_fd, buf, strlen(buf), 0, "send MAIL FROM");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv MAIL FROM");
```

9. 使用 `send_message` 函数发送 RCPT TO 命令，并打印发送的命令。并且使用 `receive_message` 函数接收邮件服务器对 RCPT TO 命令的响应，并打印在控制台上。

```
// TODO: Send RCPT TO command and print server response
sprintf(buf, "RCPT TO:<%s>\r\n", receiver);
send_message(s_fd, buf, strlen(buf), 0, "send RCPT TO");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv RCPT TO");
```

10. 使用 `send_message` 函数发送 DATA 命令，并打印发送的命令。并且使用 `receive_message` 函数接收邮件服务器对 DATA 命令的响应，并打印在控制台上。

```
// TODO: Send DATA command and print server response
const char *DATA = "DATA\r\n";
send_message(s_fd, DATA, strlen(DATA), 0, "send DATA");
receive_message(s_fd, buf, MAX_SIZE, 0, "recv DATA");
```

11. 构造邮件的消息头，包括发件人、收件人、主题等信息，并使用 `send_message` 函数发送消息头。

```
// 消息头
```

```
    if (subject) {
        sprintf(buf, "From: %s <%s>\nTo: %s\nSubject: %s\n"
            "Mime-Version: 1.0\nContent-Type: multipart/mixed; "
            "boundary=\"qwertyuiopasdfghjklzxcvbnm\"\r\n\r\n", from, user, receiver,
subject);
    } else {
        sprintf(buf, "From: %s <%s>\nTo: %s\nMime-Version: 1.0\n"
            "Content-Type: multipart/mixed; "
            "boundary=\"qwertyuiopasdfghjklzxcvbnm\"\r\n\r\n", from, user, receiver);
    }
    send_message(s_fd, buf, strlen(buf), 0, "send header");
```

12. 如果指定了邮件正文（`msg` 参数），则发送邮件正文。如果正文是文件路径，则先读取文件内容，然后使用 `send_message` 函数发送正文。

```
// 正文
    struct stat st;
    if (msg != NULL) {
        sprintf(buf, "--qwertyuiopasdfghjklzxcvbnm\r\n"
            "Content-Type: text/plain; charset=UTF-8\r\n\r\n");
        send_message(s_fd, buf, strlen(buf), 0, "send msg boundary & header");
        if (stat(msg, &st) == 0) {
            // 此时 msg 是一个文件
            FILE *fp = fopen(msg, "r");
            char *data = read_file(msg);
            send_message(s_fd, data, strlen(data), 0, "send message");
            free(data);
        } else {
            send_message(s_fd, msg, strlen(msg), 0, "send message");
        }
    }
    send_message(s_fd, "\r\n", 2, 0, "<CR><LF>");
```

13. 如果指定了附件路径（`att_path` 参数），则发送附件。先发送附件的分隔符和头部信息，然后将附件进行 Base64 编码，并使用 `send_message` 函数发送附件内容。并且使用 `send_message` 函数发送结束附件的分隔符和结束标记。

```
    // 附件
    if (att_path != NULL && stat(att_path, &st) == 0) {
        // 分隔符
        sprintf(buf, "--qwertyuiopasdfghjklzxcvbnm\r\n");
        send_message(s_fd, buf, strlen(buf), 0, "send boundary");
        // 发送附件头
        const char *file_name = get_file_name(att_path);
        sprintf(buf, "Content-Type: application/octet-stream\n"
            "Content-Disposition: attachment; "
            "filename=\"%s\"\nContent-Transfer-Encoding: base64\r\n\r\n", file_name);
        send_message(s_fd, buf, strlen(buf), 0, "send attachment header");
```

```
        // 发送附件数据
        FILE *fp = fopen(att_path, "r");
        FILE *fp64 = fopen("tmp.txt", "w");
        encode_file(fp, fp64);
        fclose(fp);
        fclose(fp64);
        char *att_data = read_file("tmp.txt");
        send_message(s_fd, att_data, strlen(att_data), 0, "send attachment");
        send_message(s_fd, "\r\n", 2, 0, "<CR><LF>");
        free(att_data);
        if (remove("tmp.txt") != 0) {
            perror("Fail to remove tmp.txt");
        }
    }
    sprintf(buf, "--qwertyuiopasdfghjklzxcvbnm\r\n");
    send_message(s_fd, buf, strlen(buf), 0, "send boundary");
```

14. 使用 `receive_message` 函数接收邮件服务器对邮件发送的响应，并打印在控制台上。

```
    // TODO: Message ends with a single period
    send_message(s_fd, end_msg, strlen(end_msg), 0, "send end msg");
    receive_message(s_fd, buf, MAX_SIZE, 0, "recv end msg");
```

15. 使用 `send_message` 函数发送 QUIT 命令，表示邮件发送结束。并且使用 `receive_message` 函数接收邮件服务器对 QUIT 命令的响应，并打印在控制台上。

```
    const char *QUIT = "QUIT\r\n";
    send_message(s_fd, QUIT, strlen(QUIT), 0, "send QUIT");
    receive_message(s_fd, buf, MAX_SIZE, 0, "recv QUIT");
```

16. 关闭套接字，结束邮件发送过程。

```
    close(s_fd);
```

2. 邮件接收客户端详细设计

1. 定义了一个常量 `MAX_SIZE`，它表示接收缓冲区的最大大小。并声明了一个全局字符数组 `buf`，用于存储接收和发送的数据。

```
#define MAX_SIZE 65535
char buf[MAX_SIZE+1];
```

2. `receive_message` 函数用于接收消息，它接受参数 `s_fd`（套接字文件描述符）、`buf`（接收缓冲区）、`size`（接收缓冲区大小）、`flags`（标志位）和 `error_message`（错误提示信息）。该函数使用 `recv` 函数从套接字中接收数据，并在发生错误时输出错误信息并终止程序。接收到的数据存储在 `buf` 中，并添加一个终止符号 `\0`。然后，函数打印接收到的消息并返回接收到的字节数。

```
int receive_message(int s_fd, void *buf, size_t size, int flags, const char*
error_message) {
    int r_size;
```

```
    if ((r_size = recv(s_fd, buf, MAX_SIZE, 0)) == -1) {
        perror(error_message);
        exit(EXIT_FAILURE);
    }
    ((char *)buf)[r_size] = '\0';
    printf("%s", (char *)buf);
    return r_size;
}
```

3. `send_message` 函数用于发送消息，它接受参数 `s_fd`（套接字文件描述符）、`buf`（待发送数据）、`size`（待发送数据大小）、`flags`（标志位）和 `message`（消息提示信息）。该函数使用 `send` 函数将数据发送到套接字，并打印发送的消息。

```
void send_message(int s_fd, const void *buf, size_t size, int flags, const char *message)
{
    printf(">>> %s\n", message);
    send(s_fd, buf, size, flags);
}
```

4. `recv_mail` 函数用于接收邮件。它首先指定邮件服务器的域名、POP3 服务器的端口号、用户名和密码等信息。然后，它通过调用 `gethostbyname` 函数从域名获取邮件服务器的 IP 地址，并将获取到的 IP 地址存储在 `dest_ip` 变量中。

```
    const char* host_name = "pop.qq.com"; // TODO: Specify the mail server domain name
    const unsigned short port = 110; // POP3 server port
    const char* user = "838818923@qq.com"; // TODO: Specify the user
    const char* pass = "tatboirwvsvsbegi"; // TODO: Specify the password
    char dest_ip[16];
    int s_fd; // socket file descriptor
    struct hostent *host;
    struct in_addr **addr_list;
    int i = 0;
    int r_size;
    // Get IP from domain name
    if ((host = gethostbyname(host_name)) == NULL)
    {
        herror("gethostbyname");
        exit(EXIT_FAILURE);
    }
    addr_list = (struct in_addr **) host->h_addr_list;
    while (addr_list[i] != NULL)
        ++i;
    strcpy(dest_ip, inet_ntoa(*addr_list[i-1]));
```

5. 接下来，函数创建一个套接字，并通过调用 `connect` 函数与邮件服务器建立 TCP 连接。

```
    if ((s_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(EXIT_FAILURE);
```

```
}
struct sockaddr_in socket_address;
// 写入 protocol，端口号和 IP 地址
socket_address.sin_family = AF_INET;
socket_address.sin_port = htons(port);
socket_address.sin_addr.s_addr = inet_addr(dest_ip);
bzero(&(socket_address.sin_zero), 8);
connect(s_fd, (struct sockaddr*) &socket_address, sizeof(struct sockaddr));
```

6. 调用 `receive_message` 函数接收服务器的欢迎消息，并打印出来。

```
receive_message(s_fd, buf, MAX_SIZE, 0, "recv welcome");
```

7. 通过调用 `send_message` 函数发送用户名和密码，并接收服务器的响应。

```
// 用户名
sprintf(buf, "USER %s\r\n", user);
send_message(s_fd, buf, strlen(buf), 0, buf);
receive_message(s_fd, buf, MAX_SIZE, 0, "recv USER");
// 密码
sprintf(buf, "PASS %s\r\n", pass);
send_message(s_fd, buf, strlen(buf), 0, buf);
receive_message(s_fd, buf, MAX_SIZE, 0, "recv PASS");
```

8. 接着，发送 `STAT` 命令并接收服务器的响应。

```
const char *STAT = "STAT\r\n";
send_message(s_fd, STAT, strlen(STAT), 0, STAT);
receive_message(s_fd, buf, MAX_SIZE, 0, "recv STAT");
```

9. 发送 `LIST` 命令并接收服务器的响应。

```
const char *LIST = "LIST\r\n";
send_message(s_fd, LIST, strlen(LIST), 0, LIST);
receive_message(s_fd, buf, MAX_SIZE, 0, "recv LIST");
```

10. 发送 `RETR 1` 命令获取第一封邮件的内容，并打印出来。

```
const char *RETR = "RETR 1\r\n";
send_message(s_fd, RETR, strlen(RETR), 0, RETR);
r_size = receive_message(s_fd, buf, MAX_SIZE, 0, "recv RETR");
int len = atoi(buf + 4);
len -= r_size;
while (len > 0) {
    r_size = receive_message(s_fd, buf, MAX_SIZE, 0, "recv EMAIL CONTENT");
    len -= r_size;
}
```

11. 发送 `QUIT` 命令关闭与服务器的连接。

```
const char *QUIT = "QUIT\r\n";
send_message(s_fd, QUIT, strlen(QUIT), 0, QUIT);
receive_message(s_fd, buf, MAX_SIZE, 0, "recv QUIT");
```

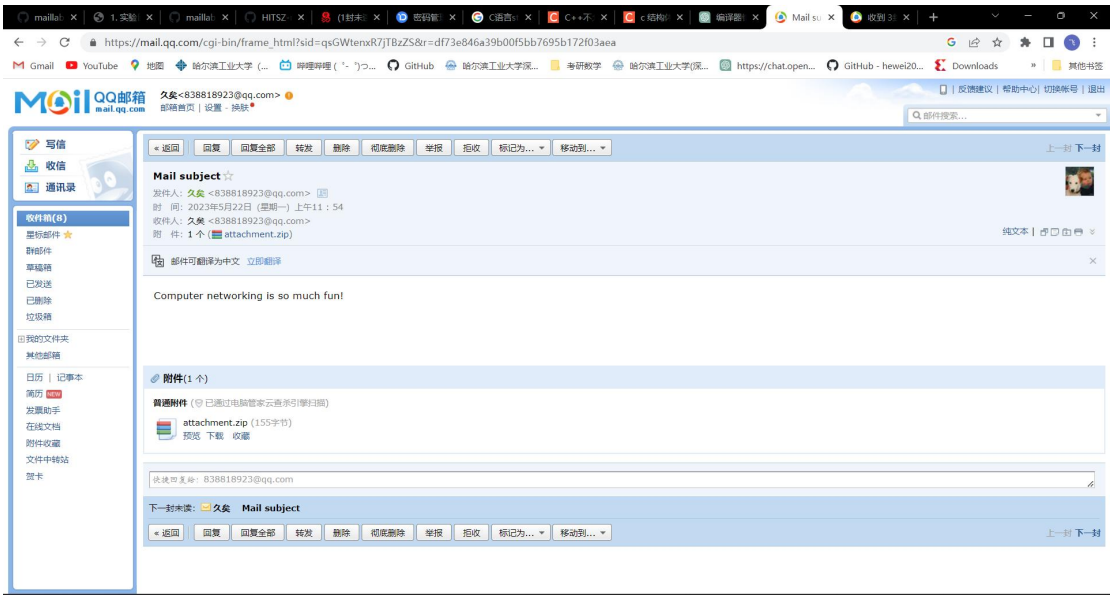
12. 最后，关闭套接字，函数执行完毕。

```
close(s_fd);
```

二、 实验结果截图及分析

（对你自己实验的测试结果进行评价）

1. 邮件发送客户端实验结果及分析



可以看到邮箱正确接收了正文和附件内容

2. 邮件接收客户端实验结果及分析

200110625@comp3:~/net_lab/lab_6/maillab$ ./recv
+OK XMail POP3 Server v1.0 Service Ready(XMail v1.0)
>>> USER 838818923@qq.com

+OK
>>> PASS tatboirwvsvsbegi

+OK
>>> STAT

+OK 6 28215
>>> LIST

+OK
1 4146
2 3934
3 7398
4 8529
5 2088

6 2120
.
>>> RETR 1

+OK 4146
X-QQ-XMAILINFO:
NnBlO8MsmACrVeBnEYVR0/8bjH2P02Go8HOKKnrpVR3SnMoUXmGjKn6PK9qmvQ

Zxegv/PGB2YTfZx60a7PCoUtmSQhfgsAkbP8cZaTEf+qT/BD7/W07Wjq/kCgi+KOD3dspo
nbTjfEZ

7HYKDv96SrxgrkFQMc4schYQlpiwoO6+obsycEFyTaiqOctZK7NUTgh7DqQ9IxnZVpBfC
Kgqjep1V

nAyJ0ID73ODcUv+mBTy2rsei7ArXbOz3ZY3ao60FwdApT81eBKfGcUNJg5T3u/uwMWm
4YYxgihbz6

6py9KnWM56N92TKi20kGaPUCXVutpKRKskiHlpUyT3AGbx0wn36s4S0M11ESIkUp4+2
4d5pa3c1T/

SCGaq84WGlNnhw9N0s06vWqqh+g2CdxKW5EfVn7yYZOlI8iUdwvOkyO20WBb07bePV
6en6XmtaeJQ

U8Bf61g7yHFDLSSTjzJc+58w4oVEqoQAyxOhLvK5wGv4bA6WrSDeUlPvyLDHBPpcXg
MpdIWAescZR

8MQsPt6c/ZhsWhPqa3mdAqu5sRbhvYM8au/6BwkcFTB3k+xKjJLlxAt6s4LUGXw6kVyTZ
Y7JN4Ba6

6MDWDWd+15DLY7Oww/Fp17jfY0+qfryhhQBeznZEC8YHKzC9snIHm6PhWM3Sq7mlK
I7WWnNGpAKuf

x8Edu/z4u/tQbvyW6Sl4vHFoMvFrY6pVGI2JvlJq3YIKjGqY+k108PhleURfZRQHO6RnCw
YHtNCK7

Szo/pVXty2nxPvaqWwHdlr54A3PmK4u40thtuUu3FR8NnSD3hldLSjT4oICIKOB0mHrphX
SVpcl6S

HGLZSfABibl5RhNEeFByfsW9x7Xxo1DK4cfRFW3qbe40YQMrAoHgLXT21gxgRvu3frc
VJWjfdLAka

gR4/WsofJcdYf93+ljGX7i2arsqJjEhYE4vf107Wxt1y/Si9Wb2UxeqFFL0u6pHsAjfdhchdHg=
X-QQ-INNER-SUBJECT-ID: 19771
X-QQ-INNER-ID: 19771

From: "=?utf-8?B?UVHpn7PkuZDpobnnm67nu4Q=?=" <music@tencent.com>
To: "=?utf-8?B?ODM4ODE4OTIz?=" <838818923@qq.com>
Subject: =?utf-8?B?5oGt5Zac5L2g77yM6LGq5Y2O57u/6ZK757ut6LS5?=
 =?utf-8?B?5oiQ5Yqf?=
Mime-Version: 1.0
Content-Type: multipart/alternative;
	boundary="----=_NextPart_6458BB0F_44F50120_5DF2456E"
Content-Transfer-Encoding: 8Bit
Date: Mon, 8 May 2023 17:04:15 +0800
X-Priority: 3
Message-ID: <tencent_F5720E32E94A831C636F4505C2FF979B6407@qq.com>
X-QQ-MIME: TCMime 1.0 by Tencent
X-Mailer: QQMail 2.x
X-QQ-Mailer: QQMail 2.x
X-QQ-STYLE: 1
X-QQ-mid: sysmailb50t1683536655tg0uilpk5

This is a multi-part message in MIME format.

------=_NextPart_6458BB0F_44F50120_5DF2456E
Content-Type: text/plain;
	charset="utf-8"
Content-Transfer-Encoding: base64

5oGt5Zac5L2g77yM6LGq5Y2O57u/6ZK757ut6LS55oiQ5YqfKDIwMjMtMDUtMDgmbmJz
cDsx
NzowNDoxNSkNCiAgICAgICAgICAgICANCiAgICAgICAgICAgICAgICAg5L2g5LqOMj
AyMy0w
NS0wOCZuYnNwOzE3OjA0OjE157ut6LS5MeW5tOixquWNjue7v+mSu++8jOacieaViOacn
+iH
szIwMjQtMDUtMTQmbmJzcDsxNzowNDoxNeOAgg0KICAgICAgICAgICAgIA0KICAgI
CAgICAg
ICAgICAgICDnmb7pobnkuJPkuqvnibnmnYPvvIzov5jmnInlpJrlsJHmnKrkvb/nlKjvvJ/k
u4rml6XmjqojZDkvJrlkZjmm7LlupPjgIHkuIvovb3nibnmnYPjgIHkuKrmgKfkuLvpopjn
rYnotoXlpJrnibnmnYPjgILlhbPms6giUVHpn7PkuZBWSVAi5b6u5L+h5YWs5LyX5Y+377y
M
5piO5pif5ZGo6L6544CB5LiT5bGe5oq15omj5Yi444CB5ryU5ZSx5Lya6Zeo56Wo44CB55S
1
5b2x56Wo562J56aP5Yip77yM5oqi5YWI6aKG77yBDQogICAgICAgICAgICAg6L+b5YWl
5a6Y
572R

------=_NextPart_6458BB0F_44F50120_5DF2456E
Content-Type: text/html;

charset="utf-8"
Content-Transfer-Encoding: base64

IDxkaXYgY2xhc3M9Im9wZW5fZW1haWwgY2xlYXJmaXgiIHN0eWxlPSJtYXJnaW4tbG
VmdDog
OHB4OyBtYXJnaW4tdG9wOiA4cHg7IG1hcmdpbi1ib3R0b206IDhweDsgbWFyZ2luLXJpZ
2h0
OiA4cHg7Ij4NCiAgICAgICAgPGRpdiBjbGFzcz0iaW1nX2xlZnQiPg0KICAgICAgICAg
PGEgaHJlZj0iaHR0cDovL20ucXpvbmUuY29tL2w/Zz0yODI3Ij4NCiAgICAgICAgICAg
ICA8aW1nIHNyYz0iaHR0cDovL3kuZ3RpbWcuY24vbXVzaWMvcGhvdG9fbmV3L1QwM
TFNMDAw
MDAxR3VhNGQyUTZuTUkuanBnIiBoZWlnaHQ9IjkwIg0KICAgICAgICAgICAgICAg
aWR0aD0iMTI1IiAvPjwvYT48L2Rpdj4NCiAgICAgICAgPGRpdiBjbGFzcz0id29yZF9yaW
do
dCI+DQoNCiAgICAgICAgICAgIDxoNT4NCiAgICAgICAgICAgICAgICAgICAgICAgICDmga3llpzkvaDv
vIzo
sarljY7nu7/pkrvnu63otLnmiJDlip88c3Bhbj4oMjAyMy0wNS0wOCZuYnNwOzE3OjA0OjE1
KTwvc3Bhbj48L2g1Pg0KDQogICAgICAgICAgICA8aDY+DQogICAgICAgICAgICAgICAg
g5L2g
5LqOMjAyMy0wNS0wOCZuYnNwOzE3OjA0OjE157ut6LS5MeW5tOixquWNjue7v+mSu
++8jOac
ieaViOacn+iHszIwMjQtMDUtMTQmbmJzcDsxNzowNDoxNeOAgjwvaDY+DQogICAgICAg
AgICAg
ICA8cD4NCiAgICAgICAgICAgICAgICAgICAgICAgICDnmb7pobnkuJPkuqvnibnmnYPvvIzov5jmnInlpJrl
sJHmnKrkvb/nlKjvvJ/ku4rml6XmjqjojZDkvJrlkZjmm7LlupPjgIHkuIvovb3nibnmnYPj
gIHkuKrmgKfkuLvpopjnrYnotoXlpJrnibnmnYPjgILlhbPms6giUVHpn7PkuZBWSVAi5b6u
5L+h5YWs5LyX5Y+377yM5piO5pif5ZGo6L6544CB5LiT5bGe5oq15omj5Yi444CB5ryU5
Sx
5Lya6Zeo56Wo44CB55S15b2x56Wo562J56aP5Yip77yM5oqi5YWI6aKG77yBPC9wPg0KI
CAg
ICAgICAgICAgPGEgY2xhc3M9ImVtYWlsX2J0biIgdGFyZ2V0PSJfYmxhbmsiIGhyZWY9I
mh0
dHA6Ly9tLnF6b25lLmNvbS8iPui/m+WFpeWumOe9kTwvYT48L2Rpdj4NCiAgICA8L2Rp
dj4N
Cg==

------=_NextPart_6458BB0F_44F50120_5DF2456E--

.
>>> QUIT

+OK Bye

可以看到 recv 函数正确的获取了邮箱中第一封邮件，函数工作正常。

三、 实验中遇到的问题及解决方法

（包括设计过程中的错误及测试过程中遇到的问题）

我在整个代码编写的过程中共遇到两个问题。
1. 在写 send.c 时，对于正文内容的理解不清楚，忘记加上末尾的结束字符串。所以在发送和读取的过程中总是出现问题。
解决措施：重新认真阅读指导书发现了这个问题并顺利解决
2. 一开始采用网易邮箱，总是无法正确通过 authentication，后面切换成 QQ 邮箱正常工作
解决措施：这个我也不知道为啥，可能网易邮箱本身有强大的安全措施吧。。。。

四、 实验收获和建议

（关于本学期计算机网络实验的三种类型：配置验证实验、协议栈系列实验、Socket 编程实验，请给出您对于这三种类型实验的收获与体会，给出评论以及改进的建议。）

配置验证实验：
这部分实验从实验设计来说是非常好的，但是纵观整个课程实验，我个人认为 9 个实验对于同学们来说还是压力蛮大的，而这部分验证实验相比于另外两个实验来说，对于同学们的提升效果较弱，因此我认为可以减轻这部分实验的比重

协议栈实验：
我认为协议栈实验的框架代码和本身的内容安排是极为优秀的，既贴合了课内知识，又锻炼了计算机学生的动手操作能力，这部分实验设计的非常好。

Socket 编程实验：
这个最后的实验说实话我认为难度相比于之前的实验来说，上了一个大台阶，而指导书相对于前面的部分反而更加精简，这给我在写这个实验的过程中带来了很大的难题。因此我认为最后这个实验的指导书可能需要更加细致一点，方便同学们更好的上手这个实验。

总体来说，这学期的计算机网络课程实验设计的非常优秀，感谢实验老师和助教一学期的辛苦付出！！！