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

实验报告

开课学期：2024 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：智能强基-计算机

学生学号：210010101

学生姓名：房煊梓

评阅教师：

报告成绩：

实验与创新实践教育中心制

2024 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

邮件发送客户端需要完成 send.c，使它能发送带有附件的邮件。以下为具体实现：

(1)填写 send_mail 函数开头的信息

根据 TO DO 的提示填写信息，包括邮箱服务器域名（这里选用 smtp.163.com），SMTP 服务器端口 25，用户名，授权码和发件人邮箱地址。

```
const char* end_msg = "\r\n.\r\n";
const char* host_name = "smtp.163.com";     // TODO: Specify the mail server domain name
const unsigned short port = 25;             // SMTP server port
const char* user = "**************";      // TODO: Specify the user
const char* pass = "****************";      // TODO: Specify the password
const char* from = "*************@163.com"; // TODO: Specify the mail address of the sender
```

(2)创建 socket 并建立到邮箱服务器的 TCP 连接

首先调用 socket 函数函数创建并初始化一个 socket 接口。其中 domain 参数在使用 IPv4 协议时为 AF_INET，type 参数在邮件客户端实验中为 SOCK_STREAM，protocol 参数设置为 0。将返回的描述符赋给 s_fd。

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

然后调用 connect 函数向邮件服务器发送 TCP 连接请求。在调用之前需要使用 sockaddr_in 结构体构造好 servaddr 参数，其中 sin_family 必须为 AF_INET，sin_port 是原始的 port 经过 htons 函数进行大小端转换得到，sin_addr 的 s_addr 是原始的 dest_ip 经过 inet_addr 函数转换为 32 位二进制网络字节序的 IPv4 地址，sin_zero 使用 bzero 函数填充 0。servaddr 参数准备好之后再调用 connect 函数发送 TCP 连接请求，其中 servaddr 传参时需要进行强制转换。

```
// and establish a TCP connection to the mail server
// servaddr参数,由sockaddr_in结构体表示
struct sockaddr_in servaddr;
// 地址族,IP协议必须为AF_INET
servaddr.sin_family = AF_INET;
// 端口号,注意大小端转换
servaddr.sin_port = htons(port);
// IP地址,需要使用inet_addr函数转换成32位二进制网络字节序的IPv4地址
servaddr.sin_addr.s_addr = inet_addr(dest_ip);
// 使用bzero函数填充0
bzero(&(servaddr.sin_zero),sizeof(servaddr.sin_zero));
// 发送连接请求,注意强制转换
if((connect(s_fd, (struct sockaddr *)&servaddr, sizeof(servaddr))) == -1) {
    herror("connect");
    exit(EXIT_FAILURE);
}
```

(3)发送 EHLO 命令并打印服务器响应

由于 EHLO 命令的格式为 EHLO <domain><CR><LF>，且本实验采用的邮箱为 163 邮箱，故填充 EHLO 命令：EHLO 163.com\r\n，然后打印命令信息，再调用 send 函数发送命令，最后调用 print_response 函数打印服务器的响应。

```
// Send EHLO command and print server response
const char* EHLO = "EHLO 163.com\r\n"; // TODO: Enter EHLO command here
strcpy(buf, EHLO);
// 打印信息
printf("%s",EHLO);
// 发送
send(s_fd, EHLO, strlen(EHLO), 0);
// TODO: Print server response to EHLO command
// 做法参考前面的Print welcome message
print_response(s_fd);
```

print_response 函数用于打印服务器响应，其实现参考了前面代码的 Print welcome message 的做法，将打印响应的动作写成一个函数，方便多次调用。

```
// 将打印响应的动作写成一个函数，方便多次调用。做法参考Print welcome message。
void print_response(int s_fd)
{
    int r_size = 0;
    if ((r_size = recv(s_fd, buf, MAX_SIZE, 0)) == -1){
        perror("response");
        exit(EXIT_FAILURE);
    }
    buf[r_size] = '\0';
    printf("%s", buf);
}
```

(4)进行身份验证并打印服务器响应

身份验证主要有以下三个步骤：

①发送 AUTH login 命令。由于 AUTH 命令的格式为 AUTH <para><CR><LF>，且本实验采用回应的认证方式为 login，故填充命令：AUTH login\r\n，然后打印命令信息，发送命令，最后打印服务器响应。

```
// 1.AUTH login命令,做法类似于EHLO
const char* AUTH_login = "AUTH login\r\n"; // TODO: Enter EHLO command here
strcpy(buf, AUTH_login);
// 打印信息
printf("%s",AUTH_login);
// 发送
send(s_fd, AUTH_login, strlen(AUTH_login), 0);
// TODO: Print server response to AUTH_login command
print_response(s_fd);
```

②输入邮箱名。注意需要调用 encode_str 函数对原始的邮箱名进行 base64 编码之后才能发送。

```
// 2.输入邮箱名,需要base64编码,并且需要打印服务器响应
strcpy(buf,encode_str(user));
// 打印信息
printf("%s",buf);
// 发送
send(s_fd, buf, strlen(buf), 0);
print_response(s_fd);
```

③输入授权码。注意需要调用 encode_str 函数对原始的授权码进行 base64 编码之后才能发送。

```
// 3.输入授权码,需要base64编码,并且需要打印服务器响应
strcpy(buf,encode_str(pass));
// 打印信息
printf("%s",buf);
// 发送
send(s_fd, buf, strlen(buf), 0);
print_response(s_fd);
```

(5)发送 MAIL FROM 命令并打印服务器响应

由于 MAIL FROM 命令的格式为 MAIL FORM:<reverse-path><CR><LF>，其中的 reverse-path 为发件人的邮箱地址即步骤(1)中填写的 from，故通过 strcat 函数追加字符串的方式来完成相应格式，构造 MAIL FROM 命令。然后打印命令信息，发送命令，最后打印服务器响应。

```
// TODO: Send MAIL FROM command and print server response
// 通过strcat追加字符串来完成相应格式
strcpy(buf,"Mail FROM:<");
strcat(buf,from);
strcat(buf,">\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(6)发送 RCPT TO 命令并打印服务器响应

由于 RCPT TO 命令的格式为 RCPT TO:<forward-path><CR><LF>，其中的 forward-path 为接收者的邮箱地址即传入 send_mail 函数的 receiver 参数，故同样采用 strcat 函数追加字符串的方式构造 RECP TO 命令，然后打印命令信息，发送命令，最后打印服务器响应。

```
// TODO: Send RCPT TO command and print server response
// 做法与MAIL FROM类似
strcpy(buf,"Rcpt TO:<");
strcat(buf,receiver);
strcat(buf,">\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(7)发送 DATA 命令并打印服务器响应

由于 DATA 命令的格式为 DATA<CR><LF>，故填充命令：data\r\n，然后打印命令信息，发送命令，最后打印服务器响应。

```
// TODO: Send DATA command and print server response
strcpy(buf,"data\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(8)发送邮件信息

发送邮件信息主要有以下四个步骤：

①填写和发送邮件头

填写写信人邮件地址，收信人邮件地址，MIME 版本，内容的类型和格式（这里为 multipart/mixed，表示多种类型的消息）之后加入分割边界，然后再填写邮件主题。填写好之后打印信息并发送。

```
// TODO: Send message data
// 填写邮件头
// From:写信人邮件地址
strcpy(buf,"From:");
strcat(buf,from);
strcat(buf,"\r\n");
// To:收信人邮件地址
strcat(buf,"To:");
strcat(buf,receiver);
strcat(buf,"\r\n");
// MIME版本
strcat(buf,"MIME-Version: 1.0");
strcat(buf,"\r\n");
// 内容的类型和格式
strcat(buf,"Content-Type: multipart/mixed; ");
// 分割边界
strcat(buf,"boundary=qwertyuiopasdfghjklzxcvbnm");
strcat(buf,"\r\n");
// Subject:邮件主题
strcat(buf,"Subject:");
strcat(buf,subject);
strcat(buf,"\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
```

②判断是否有正文消息，若有则发送

根据传入 send_mail 函数的 msg 参数判断是否有正文消息：若 msg 为空则没有正文消息，跳过此步骤；若 msg 不为空则有正文消息，需要进行发送。

```
// 如果消息部分不为空
if(msg != NULL) {
```

首先加入分割边界，然后填写内容的类型和格式（这里为 text/plain，表示文本），再打印信息并发送。

```
// 分割边界
strcpy(buf,"\r\n");
strcat(buf,"--qwertyuiopasdfghjklzxcvbnm");
strcat(buf,"\r\n");
// 内容的类型和格式
strcat(buf,"Content-Type: text/plain");
strcat(buf,"\r\n\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
```

由于发送正文消息的方式可能是直接在命令中填写正文消息，也可能是通过类似 message.txt 文件的形式表示正文，所以需要分类处理。调用 fopen 函数尝试打开 msg 指向的文件，若不为空则说明其为正文内容的路径，否则为消息本身。

```
// 消息内容
FILE*message = fopen(msg,"r");
```

对于 msg 为正文内容的路径的情况，调用 fread 函数读取其中内容并添加结束符和\r\n，再打印信息并发送，最后关闭文件。

```
if(message != NULL) {
    // 若message不为空则说明msg是文件路径,需要从中读取消息内容
    int size = fread(buf,1,MAX_SIZE,message);
    buf[size] = '\0';
    strcat(buf,"\r\n");
    // 打印信息
    printf("%s",buf);
    // 发送
    send(s_fd,buf,strlen(buf),0);
    fclose(message);
```

对于 msg 为消息本身的情况，则添加\r\n 后打印信息并发送。

```
} else {
    // 若message为空则说明msg是消息本身
    strcpy(buf,msg);
    strcat(buf,"\r\n");
    // 打印信息
    printf("%s",buf);
    // 发送
    send(s_fd,buf,strlen(buf),0);
}
```

③判断是否有附件，若有则发送

根据传入 send_mail 函数的 att_path 参数判断是否有附件：若 att_path 为空则没有附件，跳过此步骤；若 att_path 不为空则有附件，需要进行发送。

```
// 如果附件路径不为空,则说明需要发送附件
if(att_path != NULL) {
```

首先加入分割边界，然后填写内容的类型和格式（这里为 application/octet-stream，可以表示邮件附件），下载的文件名 filename（这里为 att_path），使用编码（这里为 base64），再打印信息并发送。

```
// 分割边界
strcpy(buf, "\r\n");
strcat(buf, "--qwertyuiopasdfghjklzxcvbnm");
strcat(buf, "\r\n");
// 邮件附件可以用application/octet-stream子类型表示
strcat(buf, "Content-Type: application/octet-stream");
strcat(buf, "\r\n");
// 附件名
strcat(buf, "Content-Disposition: attachment; filename=");
strcat(buf, att_path);
strcat(buf, "\r\n");
// 编码:base64
strcat(buf, "Content-Transfer-Encoding: base64");
strcat(buf, "\r\n\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd, buf, strlen(buf), 0);
```

接下来需要发送附件内容。首先，调用 encode_file 函数对原始附件 att_origin 进行 base64 编码，保存在 att_base64 中。

```
// 对原始附件att_origin进行base64编码,保存在att_base64中
FILE *att_origin = fopen(att_path,"rb");
FILE *att_base64 = tmpfile();
encode_file(att_origin,att_base64);
fclose(att_origin);
```

然后调用 fseek 和 ftell 函数获取编码后的附件的大小，调用 malloc 函数分配相应的内存。

```
// 根据att_base64的大小来分配内存
fseek(att_base64,0,SEEK_END);
int att_size = ftell(att_base64);
char*att_content = (char*)malloc((att_size+1)*sizeof(char));
```

再调用 rewind 函数将指针重置到文件的开头，调用 fread 函数读取附件的内容保存到 att_content 中，最后发送附件内容。

```
// 重置指针
rewind(att_base64);
// 读取附件内容并保存
fread(att_content,1,att_size,att_base64);
att_content[att_size] = '\0';
fclose(att_base64);
// 发送
send(s_fd,att_content,strlen(att_content),0);
```

④发送单个句点表示消息结束，并打印服务器响应
由于结束标志字符串 end_msg 已经由代码框架直接给出，故填充到 buf 后打印命令信息，发送命令，最后打印服务器响应。

```
// TODO: Message ends with a single period
strcpy(buf, end_msg);
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(9)发送 QUIT 命令并打印服务器响应，并且关闭连接
由于 QUIT 命令的格式为 QUIT<CR><LF>，故填充命令：quit\r\n，然后打印命令信息，发送命令，打印服务器响应。最后，调用 close 函数关闭连接。

```
// TODO: Send QUIT command and print server response
strcpy(buf,"quit\r\n");
// 打印信息
printf("%s",buf);
// 发送
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);


close(s_fd);
```

2. 邮件接收客户端详细设计

邮件接收客户端需要完成 recv.c，使它能与服务器交互，实现获取总邮件个数及大小、每封邮件的编号及大小、第一封邮件的内容等功能。以下为具体实现：

(1)填写 recv_mail 函数开头的信息

根据 TO DO 的提示填写信息，包括邮箱服务器域名（这里选用 pop.163.com），POP3 服务器端口 110，用户名和授权码。

```
const char* host_name = "pop.163.com";       // TODO: Specify the mail server domain name
const unsigned short port = 110;             // POP3 server port
const char* user = "***********@163.com";    // TODO: Specify the user
const char* pass = "****************";      // TODO: Specify the password
```

(2)创建 socket 并建立到邮箱服务器的 TCP 连接

这里的做法与 send.c 中的做法完全一致。

```
// 这里的TO DO做法和send处的完全一致
// TODO: Create a socket,return the file descriptor to s_fd
// 参数参考socket编程简介socket函数部分
if((s_fd = socket(AF_INET,SOCK_STREAM,0)) == -1)
{
    herror("socket");
    exit(EXIT_FAILURE);
}

// and establish a TCP connection to the POP3 server
// servaddr参数,由sockaddr_in结构体表示
struct sockaddr_in servaddr;
// 地址族,IP协议必须为AF_INET
servaddr.sin_family = AF_INET;
// 端口号,注意大小端转换
servaddr.sin_port = htons(port);
// IP地址,需要使用inet_addr函数转换成32位二进制网络字节序的IPv4地址
servaddr.sin_addr.s_addr = inet_addr(dest_ip);
// 使用bzero函数填充0
bzero(&(servaddr.sin_zero),sizeof(servaddr.sin_zero));
// 发送连接请求,注意强制转换
if(connect(s_fd, (struct sockaddr *)&servaddr, sizeof(servaddr)) == -1) {
    herror("connect");
    exit(EXIT_FAILURE);
}
```

(3)发送邮箱和授权码并打印服务器响应

由于 USER 命令的格式为 USER <name><CR><LF>，其中的 name 为输入认证用户名即步骤(1)中填写的 user（不需要加密），故通过 strcat 函数追加字符串的方式来完成相应格式，构造 USER 命令。然后打印命令信息，发送命令，最后打印服务器响应。print_response 函数与 send 中的完全一致。

```
// TODO: Send user and password and print server response
// 通过strcat追加字符串来完成相应格式。注意此处邮箱和授权码都不需要加密。
// 1.邮箱
strcpy(buf,"user ");
strcat(buf,user);
strcat(buf,"\r\n");
// 打印信息
printf("%s",buf);
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

由于 PASS 命令的格式为 PASS <pass><CR><LF>，其中的 pass 为输入认证口令即步骤(1)中填写的 pass（不需要加密），故通过 strcat 函数追加字符串的方式来完成相应格式，构造 PASS 命令。然后打印命令信息，发送命令，最后打印服务器响应。print_response 函数与 send 中的完全一致。

```
//2.授权码
strcpy(buf,"pass ");
strcat(buf,pass);
strcat(buf,"\r\n");
// 打印信息
printf("%s",buf);
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(4)发送 STAT 命令并打印服务器响应

由于 STAT 命令的格式为 STAT<CR><LF>，故填充命令：stat\r\n，然后打印命令信息，发送命令，打印服务器响应。

```
// TODO: Send STAT command and print server response
strcpy(buf,"stat\r\n");
// 打印信息
printf("%s",buf);
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(5)发送 LIST 命令并打印服务器响应

由于 LIST 命令的格式为 LIST[<msg>]<CR><LF>参数可选，而这里需要获得每封邮件的编号及大小，故填充命令：list\r\n，然后打印命令信息，发送命令，打印服务器响应。

```
// TODO: Send LIST command and print server response
strcpy(buf,"list\r\n");
// 打印信息
printf("%s",buf);
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(6)检索第一封邮件并打印其内容

由于 RETR 命令的格式为 RETR msg<CR><LF>，而这里需要打印第一封邮件的内容，故填充命令：retr 1\r\n，然后打印命令信息，发送命令，打印服务器响应。

```
// TODO: Retrieve the first mail and print its content
strcpy(buf,"retr 1\r\n");
// 打印信息
printf("%s",buf);
send(s_fd,buf,strlen(buf),0);
print_response(s_fd);
```

(7)发送 QUIT 命令并打印服务器响应，并且关闭连接

由于 QUIT 命令的格式为 QUIT<CR><LF>，故填充命令：quit\r\n，然后打印命令信息，发送命令，打印服务器响应。最后，调用 close 函数关闭连接。

```
// TODO: Send QUIT command and print server response
strcpy(buf,"quit\r\n");
send(s_fd,buf,strlen(buf),0);
// 打印信息
printf("%s",buf);
print_response(s_fd);


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

编译好程序后，运行发送命令：./send -s "hello" -m message.txt -a "attachment.zip" wq87c5e7q16@163.com。其中 message.txt 的内容为：Computer networking is so much fun!而 attachment.zip 内含有一个 attachment.txt 文件，内容与 message.txt 的内容一样。

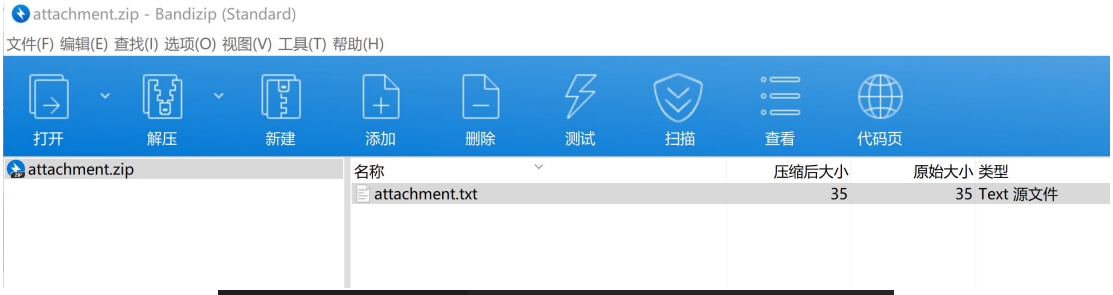
交互过程中服务器的回复信息截图如下。可以看到 EHLO，Authentication 过程，Mail FROM、Rcpt TO、data 指令，邮件内容等方面的回复信息均符合预期。

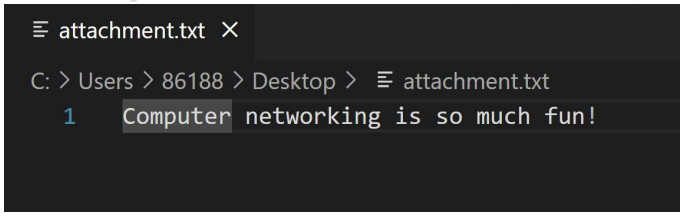
```
[06/09/24]seed@VM:~/maillab$ ./send -s "hello" -m message.txt -a "attachment.zip" wq87c5e7q16@163.com
220 163.com Anti-spam GT for Coremail System (163com[20141201])
EHLO 163.com
250-mail
250-PIPELINING
250-AUTH LOGIN PLAIN XOAUTH2
250-AUTH=LOGIN PLAIN XOAUTH2
250-coremail 1Uxr2xKj7kG0xkI17xGrU7I0s8FY2U3Uj8Cz28x1UUUUU7Ic2I0Y2Ur1S56yUCa0xDrUUUUj
250-STARTTLS
250-ID
250 8BITMIME
AUTH login
334 dXNlcm5hbWU6
cTc2ZXN0NWNxNzl3ZA==
334 UGFzc3dvcmQ6
QUZET1lKSVNVVU9CR0lDRQ==
235 Authentication successful
Mail FROM:<q76est5cq79wd@163.com>
250 Mail OK
Rcpt TO:<wq87c5e7q16@163.com>
250 Mail OK
data
354 End data with <CR><LF>.<CR><LF>
From:q76est5cq79wd@163.com
To:wq87c5e7q16@163.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiopasdfghjklzxcvbnm
Subject:hello

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/plain

Computer networking is so much fun!

--qwertyuiopasdfghjklzxcvbnm
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=attachment.zip
Content-Transfer-Encoding: base64


.
250 Mail OK queued as gzga-smtp-mta-g1-2,_____wDnl4Sar2Vmnp65GQ--.41286S2 1717940123
quit
221 Bye
[06/09/24]seed@VM:~/maillab$
```

打开接收邮件的邮箱，发现能收到邮件，且发件人和标题均符合预期。截图如下。

进入邮件查看，发现发件人和收件人均符合预期，邮件正文内容与 message.txt 内容一致，并且带有附件 attachment.zip。截图如下。

下载附件，解压后进行查看，发现附件内容符合预期。截图如下。





根据以上分析可知 send.c 能够正确发送带有附件的邮件。

2. 邮件接收客户端实验结果及分析

编译好程序后，运行接收命令./recv。
交互过程中服务器的回复信息截图如下。可以看到用户名、授权码以及 stat、list、retr 1、quit 等命令的回复信息均符合预期。

```
210010101@comp2:~/maillab$ ./recv
+OK POP3 ready
user wq87c5e7q16@163.com
+OK
pass KVAKTIPNEUNREMIQ
+OK 3 message(s) [3785 byte(s)]
stat
+OK 3 3785
list
+OK 3 3785
1 1263
2 1259
3 1263
.
retr 1
+OK 1263 octets
Received: from 163.com (unknown [58.250.172.33])
        by gzga-smtp-mta-g3-4 (Coremail) with SMTP id _____wDn7ywIqWVm9POoDA--.27308S2;
        Sun, 09 Jun 2024 21:07:20 +0800 (CST)
From:q76est5cq79wd@163.com
To:wq87c5e7q16@163.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiopasdfghjklzxcvbnm
Subject:hello
X-CM-TRANSID:_____wDn7ywIqWVm9POoDA--.27308S2
Message-Id:<6665A90A.119590.00046@m16.mail.163.com>
X-Coremail-Antispam: 1Uf129KBjDUn29KB7ZKAUJUUUUU529EdanIXcx71UUUUU7v73
        VFW2AGmfu7bjvjm3AaLaJ3UbIYCTnIWIevJa73UjIFyTuYvjxUagyCDUUUU
X-Originating-IP: [58.250.172.33]
Date: Sun, 9 Jun 2024 21:07:22 +0800 (CST)
X-CM-SenderInfo: ztxwv23wvf1lwzzgqiywtou0bp/1tbiLgr4zmV4Jc8htgAAsy

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/plain

Computer networking is so much fun!

--qwertyuiopasdfghjklzxcvbnm
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=attachment.zip
Content-Transfer-Encoding: base64

UEsDBAoAAAAAAAAAIQDPgBZ4IwAAACMAAAAOABwAYXR0YWNobWVudC50eHRVVAkAA9CazRKg
pmVmdXgLAAEE6AMAAAToAwAAQ29tcHV0ZXIgbmV0d29ya2luZyBpcyBzbyBtdWNoIGZ1biFQ
SwECHgMKAAAAAAAAACEAz4AWeCMAAAAjAAAADgAYAAAAAAAABAAAAtIEAAAAAYXR0YWNobWVu
dC50eHRVVAUAA9CazRJ1eAsAAQToAwAABOgDAABQSwUGAAAAAAEAAQBUAAAAawAAAAA
.
quit
+OK core mail
210010101@comp2:~/maillab$
```

其中，总邮件个数及大小由 stat 命令获得：

```
stat
+OK 3 3785
```

每封邮件的编号及大小由 list 命令获得：

```
list
+OK 3 3785
1 1263
2 1259
3 1263
.
```

第一封邮件的内容由 retr 1 命令获得：

```
retr 1
+OK 1263 octets
Received: from 163.com (unknown [58.250.172.33])
        by gzga-smtp-mta-g3-4 (Coremail) with SMTP id _____wDn7ywIqWVm9POoDA--.27308S2;
        Sun, 09 Jun 2024 21:07:20 +0800 (CST)
From:q76est5cq79wd@163.com
To:wq87c5e7q16@163.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiopasdfghjklzxcvbnm
Subject:hello
X-CM-TRANSID:_____wDn7ywIqWVm9POoDA--.27308S2
Message-Id:<6665A90A.119590.00046@ml6.mail.163.com>
X-Coremail-Antispam: 1Uf129KBjDUn29KB7ZKAUJUUUUU529EdanIXcx71UUUUU7v73
        VFW2AGmfu7bjvjm3AaLaJ3UbIYCTnIWIevJa73UjIFyTuYvjxUagyCDUUUU
X-Originating-IP: [58.250.172.33]
Date: Sun, 9 Jun 2024 21:07:22 +0800 (CST)
X-CM-SenderInfo: ztxwv23wvf1lwzzgqiywtou0bp/1tbiLgr4zmV4Jc8htgAAsy

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/plain

Computer networking is so much fun!

--qwertyuiopasdfghjklzxcvbnm
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=attachment.zip
Content-Transfer-Encoding: base64

UEsDBAoAAAAAAAAAIQDPgBZ4IwAAACMAAAAOABwAYXR0YWNobWVudC50eHRVVAkAA9CazRKg
pmVmdXgLAAEE6AMAAAToAwAAQ29tcHV0ZXIgbmV0d29ya2luZyBpcyBzbyBtdWNoIGZ1biFQ
SwECHgMKAAAAAAAAACEAz4AWeCMAAAAjAAAADgAYAAAAAAAABAAAAtIEAAAAAYXR0YWNobWVu
dC50eHRVVAUAA9CazRJ1eAsAAQToAwAABOgDAABQSwUGAAAAAAEAAQBUAAAAawAAAAA
.
```

根据以上分析可知 recv.c 能够正确获取相关信息。

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）

•**问题 1**：测试 send 时没有打印欢迎信息，而是出现错误：connect:Unknown host。

```
connect: Unknown host
```

解决方法：由于没有打印欢迎信息，推测错误出现在打印欢迎信息之前，于是在创建 socket 和调用 connect 处分别使用 printf 语句打印信息看是否执行成功，发现 socket 成功创建但是 connect 调用失败，仔细检查 connect 之后认为没有问题，再仔细检查创建 socket 的过程，发现判断返回值是否为-1 的语句忘记加上括号了，加上括号之后重新测试，发现能够打印欢迎信息了。

```
if((s_fd = socket(AF_INET,SOCK_STREAM,0)) == -1)
```

• **问题 2**：测试 send 时出现错误：authentication failed

```
AUTH login
334 dXNlcm5hbWU6
cTc2ZXN0NWNxNzl3ZA==

334 UGFzc3dvcmQ6
535 Error: authentication failed
QUZET1lKSVNVVU9CR0lDRQ==

502 Error: command not implemented
```

解决方法：观察打印的信息，发现输入用户名和授权码的部分均多出了一行，猜测可能是在输入用户名和授权码时错误地追加了\r\n 从而导致的错误，于是删减之后重新测试，发现能够成功 login 了。

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。）

对于配置验证实验，难度较为简单，需要认真预习和协调好团队合作，并且需要小心操作失误。配置验证实验让我熟悉了路由器、交换机等设备的工作过程以及数据报的具体传输方法，同时也提高了我的团队协作能力。

对于配置验证实验的建议是可以对一些常见的错误进行相关说明，方便同学们修正错误。

对于协议栈系列实验，难度较大，不仅需要认真阅读指导书的内容以理解协议栈的原理并按步骤完成代码，而且需要对比实际结果和 demo 来排查代码问题。协议栈实验一方面让我在实际的代码编写中对各层的数据报结构和传输过程有了更加深刻的理解，另一方面让我在不断比对分析中提高了对数据报的分析能力和排查错误的能力。

对于协议栈实验的建议是可以多分配一些课时给各个协议的实现，感觉两次课写完协议栈的压力有点大；还可以完善一下测试代码或者对测试的结果进行更加详细的说明，防止前面代码出现的问题到后面的测试才显现从而出现误导，比如我的 IP 协议测试通过了但是 ICMP 测试没有通过，初期以为不是 IP 的问题而是 ICMP 的问题，但是修改后并没有效果，经过进一步排查之后才发现是 IP 方面出现了问题。

对于 socket 编程实验，难度适中，代码量与协议栈相比较少。socket 编程实验一方面让我对 SMTP 协议和 POP3 协议的命令有了更加深刻的理解，另一方面让我在通过代码实现实际的邮件收发中收获了乐趣。

对于 socket 编程实验的建议是增加除了 QQ 邮箱以外的一些其他常见邮箱的 POP3/STMP 服务开启方法和授权码获取方法。为了避免被封号，我申请了新的 QQ 邮箱，但是发现需要申请满 14 天才能开启相关服务而这个时间显然超出了完成实验所预期的时间，并且它的界面和指导书上的版本不一样，没办法按照指导书的提示开启服务。于是申请别的邮箱来进行实验，故希望能增加其他邮箱的相关说明。

总之，本学期的计算机网络实验内容丰富，将理论课内容与实际相结合，加深了我对相关知识的理解，增强了我的代码能力和实践能力。