



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: 5 班
学生学号: 200110513
学生姓名: 宗晴
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

该实验要求编写一个简单的邮件发送客户端，将邮件发送给任意收件人。客户端需要连接到邮件发送服务器，使用 SMTP 协议进行交互。

具体来说，我们需要完成 send.c 文件中的 send_mail 函数，该函数的主要功能包括 socket 的创建与连接、消息的收发、各类邮件的发送（输入可以是邮件内容本身、包括邮件内容的文件路径，以及附件路径）。

下面阐述该函数的具体算法实现。

首先，我们需要补充完整主机名、用户名、授权码和邮件发送者邮箱等信息，其中主机名为 smtp.qq.com，如下图所示：

```
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 = " "; // TODO: Specify the user
const char* pass = " "; // TODO: Specify the password
const char* from = " "; // TODO: Specify the mail address of the sender
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;
```

在获取到目的 IP 地址后，我们创建 socket，创建成功后会返回对应的文件描述符 s_fd，接着建立一个到邮件服务器的 TCP 连接，相应的参数设置可参考指导书的提示。注意此处的 port 占据 2 字节，需要进行大小端转换，所以我们需要设置一个 swap 函数进行实现，如下图所示：

```
#define swap16(x) (((x)&0xFF) << 8) | (((x) >> 8) & 0xFF)
```

此外，在 connect 函数传参时，需要将 struct sockaddr_in 结构体表示的服务器地址和端口强制转换成 struct sockaddr*。

实现过程如下图所示：

```
// TODO: Create a socket, return the file descriptor to s_fd, and establish a TCP connection to the mail server
if((s_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("create");
    exit(EXIT_FAILURE);
}

struct sockaddr_in addr_in;
addr_in.sin_family = AF_INET;
addr_in.sin_port = swap16(port);
addr_in.sin_addr.s_addr = inet_addr(dest_ip);
memset(addr_in.sin_zero, 0, sizeof(addr_in.sin_zero));

if(connect(s_fd, (struct sockaddr *)&addr_in, sizeof(addr_in)) == -1) {
    perror("connect");
    exit(EXIT_FAILURE);
}
```

然后，我们需要实现信息的收发。在此之前我们实现一个辅助函数用于打印并发送信息，然后接收并打印信息，实现过程如下。我们首先实现一个接收并打印信息的函数 `print_response`，该函数会将接收到的信息保存在 `buf`（`buf` 是一个全局变量，作为输入和输出的缓冲区）中，然后在最后添加结束符并打印。然后我们实现一个打印并发送信息的函数 `send_print`，该函数会先将缓存在 `buf` 中的待发送信息打印出来，然后将信息发送出去。由于大部分情况发送信息后均会有回复，为方便起见我们直接在该函数内调用 `print_response` 接收并打印回复的信息。

```
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);
}

void send_print(int s_fd)
{
    printf("%s", buf);
    send(s_fd, buf, strlen(buf), 0);
    print_response(s_fd);
}
```

在打印完欢迎信息后，我们开始实现正式的信息交互。首先向服务器发送 `EHLO` 命令，`EHLO` 命令是 `SMTP` 邮件发送程序与 `SMTP` 邮件接收程序建立连接后必须发送的第一条 `SMTP` 命令，用于表明身份。我们将命令复制进缓冲区中，然后调用上述定义的 `send_print` 函数将待发送的信息打印并发送出去，然后接收并打印收到的信息，即从服务器收到的支持的命令。如下图所示：

```
// Send EHLO command and print server response
// TODO: Print server response to EHLO command
// TODO: Enter EHLO command here
strcpy(buf, "ehlo qq.com\r\n");
send_print(s_fd);
```

然后，客户端选择登录认证方式，本实验可选择 login。服务器发送经过 Base64 编码的字符串“Username:”，然后客户端发送经过 Base64 编码的用户名（可调用 encode_str 函数实现）。服务器发送经过 Base64 编码的字符串“Password:”，然后客户端发送经过 Base64 编码的授权码（可调用 encode_str 函数实现）。如用户名和授权码正确，则服务器提示认证成功。发送、接收与打印过程均与上述方法一致。过程如下图所示：

```
// TODO: Authentication. Server response should be printed out.
strcpy(buf, "AUTH login\r\n");
send_print(s_fd);

strcpy(buf, encode_str(user));
strcat(buf, "\r\n");
send_print(s_fd);

strcpy(buf, encode_str(pass));
strcat(buf, "\r\n");
send_print(s_fd);
```

接着，客户端指定邮件的发送人（mail from）和收件人（rcpt to），每次换行后，服务器都会提示成功。

```
// TODO: Send MAIL FROM command and print server response
strcpy(buf, "mail from:<");
strcat(buf, user);
strcat(buf, ">\r\n");
send_print(s_fd);

// TODO: Send RCPT TO command and print server response
strcpy(buf, "rcpt to:<");
strcat(buf, receiver);
strcat(buf, ">\r\n");
send_print(s_fd);
```

客户端输入 DATA 命令，服务器提示输入内容后以“.”表示消息结束。

```
// TODO: Send DATA command and print server response
strcpy(buf, "data\r\n");
send_print(s_fd);
```

之后，就可以编写要发送的邮件内容。

先填写邮件的发送者、接收者、版本号、内容类别（此处为 `multipart/mixed`，表示有多种类别）、用于分隔不同部分内容的边界、邮件主题，直接调用 `send` 和 `printf` 函数将这些信息发送出去并打印（因为此处无回复信息），如下图所示：

```
// TODO: Send message data
strcpy(buf, "From:");
strcat(buf, from);
strcat(buf, "\r\nTo:");
strcat(buf, receiver);
strcat(buf, "\r\nMIME-Version: 1.0\r\n");
strcat(buf, "Content-Type: multipart/mixed; ");
strcat(buf, "boundary=qwertyuiopasdfghjklzxcvbnm\r\n");
strcat(buf, "Subject:");
strcat(buf, subject);
strcat(buf, "\r\n");

send(s_fd, buf, strlen(buf), 0);
printf("%s", buf);
```

然后，若 `message` 不为空，则表示邮件正文存在内容。对于正文信息，本实验要求我们实现既能根据直接的正文内容，又能根据存储正文内容的文件路径发送信息。

我们根据 `message` 的不同类别进行邮件正文的填写。首先填入分割边界，然后填写内容的类别，此处为 `text/plain`。我们尝试打开 `message` 所指向的文件，若打开后不为空，则表示 `message` 存储的是包括邮件内容的文件路径，我们获取该邮件正文的长度，将内容复制进 `buf` 中，在最后添加上结束符、换行符，然后发送并打印该邮件主体，最后关闭文件；若打开 `message` 所指向的文件失败，则表示 `message` 中存储的直接就是邮件的正文内容，则将该内容复制进 `buf` 中，在最后添加换行符，发送并打印。过程如下图所示：


```

if(msg) {
    strcpy(buf, "\r\n--qwertyuiopasdfghjklzxcvbnm\r\n");
    strcat(buf, "Content-Type: text/plain\r\n\r\n");
    send(s_fd, buf, strlen(buf), 0);
    printf("%s", buf);
    FILE *file_mail_body = fopen(msg, "r");
    if(file_mail_body) {
        // msg = path to the file containing mail body
        int mail_size = fread(buf, 1, MAX_SIZE, file_mail_body);
        buf[mail_size] = '\0';
        strcat(buf, "\r\n");
        send(s_fd, buf, strlen(buf), 0);
        printf("%s", buf);
        fclose(file_mail_body);
    } else {
        // msg = content of mail body
        strcpy(buf, msg);
        strcat(buf, "\r\n");
        send(s_fd, buf, strlen(buf), 0);
        printf("%s", buf);
    }
}

```

若 att_path 不空，则表示存在待发送的附件。首先填入分割边界，然后填写内容的类别，此处为 application/octet-stream。填写附件的名称和编码的类别，将这些信息发送并打印出来。然后打开 att_path 所指向的文件，调用 encode_file 函数将其按照 base64 编码，结果存储在 tmp 中，关闭文件。然后调用 fseek 和 ftell 函数获取文件的大小，按照该大小分配存储空间。将附件的内容读取出来并存放在上述分配的存储空间中，在最后添加结束符，关闭文件后，将内容发送出去，如下图所示：

```

if(att_path) {
    // path to the attachment exist
    strcpy(buf, "\r\n--qwertyuiopasdfghjklzxcvbnm\r\n");
    strcat(buf, "Content-Type: application/octet-stream\r\n");
    strcat(buf, "Content-Disposition: attachment; name=");
    strcat(buf, att_path);
    strcat(buf, "\r\n");
    strcat(buf, "Content-Transfer-Encoding: base64\r\n\r\n");
    send(s_fd, buf, strlen(buf), 0);
    printf("%s", buf);

    // encode the att_file
    FILE *raw_file = fopen(att_path, "rb");
    FILE *base64_file = tmpfile();
    encode_file(raw_file, base64_file);
    fclose(raw_file);

    // get size of the att_file
    fseek(base64_file, 0, SEEK_END);
    int att_size = ftell(base64_file);
    char* att_content = (char*) malloc((att_size + 1) * sizeof(char));
    rewind(base64_file);

    // read the att_file content
    fread(att_content, 1, att_size, base64_file);
    att_content[att_size] = '\0';
    fclose(base64_file);

    send(s_fd, att_content, strlen(att_content), 0);
}

```

然后，客户端输入“.”表示邮件内容输入完毕，服务器提示成功：

```

// TODO: Message ends with a single period
strcpy(buf, end_msg);
send_print(s_fd);

```

最后，客户端输入 QUIT 命令断开与邮件服务器的连接，服务器提示连接中断。

```

// TODO: Send QUIT command and print server response
strcpy(buf, "quit\r\n");
send_print(s_fd);

```

2. 邮件接收客户端详细设计

该实验要求编写一个简单的邮件接收客户端，获取接收到的邮件。客户端需要连接到邮件接收服务器，使用 POP3 协议进行交互。

具体来说，我们需要完成 `recv.c` 文件中的 `recv_mail` 函数，该函数的主要功能包括 `socket` 的创建与连接、消息的收发、邮件的接收。

下面阐述该函数的具体算法实现。

首先，我们需要补充完整主机名、用户名、授权码等信息，其中主机名为 `pop.qq.com`，如下图所示：

```
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 = " "; // TODO: Specify the user
const char* pass = " "; // 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;
```

在获取到目的 IP 地址后，我们创建 `socket`，创建成功后会返回对应的文件描述符 `s_fd`，接着建立一个到 POP3 服务器的 TCP 连接，过程与 `send.c` 中类似，如下图所示，不再赘述：

```
// TODO: Create a socket, return the file descriptor to s_fd, and establish a TCP connection to the POP3 server
if((s_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("create");
    exit(EXIT_FAILURE);
}

struct sockaddr_in addr_in;
addr_in.sin_family = AF_INET;
addr_in.sin_port = swap16(port);
addr_in.sin_addr.s_addr = inet_addr(dest_ip);
memset(addr_in.sin_zero, 0, sizeof(addr_in.sin_zero));

if(connect(s_fd, (struct sockaddr *)&addr_in, sizeof(addr_in)) == -1) {
    perror("connect");
    exit(EXIT_FAILURE);
}
```

在打印完欢迎信息后，我们开始实现正式的信息交互。客户端输入用户名和密码进行认证，如果正确，服务器会返回成功信息，如下图所示：


```
// TODO: Send user and password and print server response
strcpy(buf, "user ");
strcat(buf, user);
strcat(buf, "\r\n");
send_print(s_fd);

strcpy(buf, "pass ");
strcat(buf, pass);
strcat(buf, "\r\n");
send_print(s_fd);
```

认证成功后，客户端可以输入一系列命令获取信息，如 STAT、LIST、RETR、DELE 等。这里分别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，如下图所示：

```
// TODO: Send STAT command and print server response
strcpy(buf, "stat\r\n");
send_print(s_fd);

// TODO: Send LIST command and print server response
strcpy(buf, "list\r\n");
send_print(s_fd);

// TODO: Retrieve the first mail and print its content
strcpy(buf, "retr 1\r\n");
send_print(s_fd);
```

最后，客户端发送 QUIT 命令，结束会话。

```
// TODO: Send QUIT command and print server response
strcpy(buf, "quit\r\n");
send_print(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)

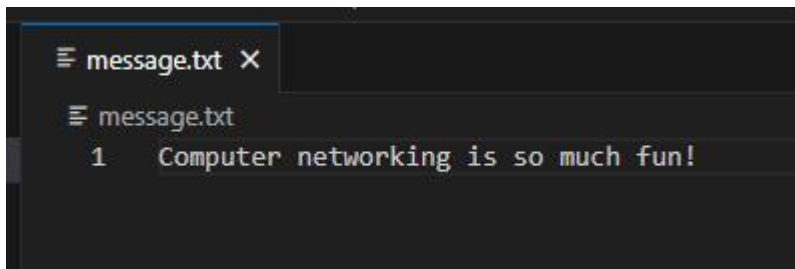
1. 邮件发送客户端实验结果及分析

命令：./send -s "Mail subject" -m message.txt -a "attachment.zip" example@example.org

主题为 Mail subject，message.txt 是邮件正文的文件路径，邮件附件为


attachment.zip

其中 message.txt 文件中存储的内容为:

A screenshot of a text editor window with a dark background. The title bar shows 'message.txt' and a close button. The editor contains a single line of text: '1 Computer networking is so much fun!'.

```
message.txt X
message.txt
1 Computer networking is so much fun!
```

attachment.zip 压缩包中存储的是 attachment.docx 文件，内容为:

 attachment.docx


hello! ↵

客户端与服务器的交互信息输出如下:

```

[05/22/23]seed@VM:~/maillab-master$ make
gcc -c base64_utils.c -o base64_utils.o
gcc -c cencode.c -o cencode.o
gcc -c cdecode.c -o cdecode.o
gcc send.c base64_utils.o cencode.o cdecode.o -o send
gcc recv.c -o recv
[05/22/23]seed@VM:~/maillab-master$ ./send -s "Mail subject" -m message.txt
-a "attachment.zip" 691745382@qq.com
220 newxmesmtplogicsvrsz12-0.qq.com XMail Esmtpl QQ Mail Server.
ehlo qq.com
250-newxmesmtplogicsvrsz12-0.qq.com
250-PIPELINING
250-SIZE 73400320
250-STARTTLS
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2
250-AUTH=LOGIN
250-MAILCOMPRESS
250 8BITMIME
AUTH login
334 VXNlcm5hbWU6
[REDACTED]

334 UGFzc3dvcmQ6
[REDACTED]

235 Authentication successful
mail from:<691745382@qq.com>
250 OK
rcpt to:<691745382@qq.com>
250 OK
data
354 End data with <CR><LF>.<CR><LF>.
From:691745382@qq.com
To:691745382@qq.com

354 End data with <CR><LF>.<CR><LF>.
From:691745382@qq.com
To:691745382@qq.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiopasdfghjklzxcvbnm
Subject:Mail subject

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/plain

Computer networking is so much fun!

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

.
250 OK: queued as.
quit
221 Bye.

```

交互过程符合预期。

打开对应的 qq 邮箱，发现成功收到了该邮件：



邮件主题为 Mail subject，邮件正文是 message.txt 文件中存储的内容，邮件附件为 attachment.zip，内容与发送端一致，符合预期：



Computer networking is so much fun!



Wireshark 抓包结果如下，均符合预期：

smtp						
No.	Time	Source	Destination	Protocol	Length	Info
219	16.109208	183.47.101.192	10.251.137.55	SMTP	119 S:	220 newxmesmtplgicsvrszb1-0.qq.com XMail Esmtpl QQ
220	16.110730	10.251.137.55	183.47.101.192	SMTP	67 C:	ehlo qq.com
222	16.125267	183.47.101.192	10.251.137.55	SMTP	225 S:	250-newxmesmtplgicsvrszb1-0.qq.com PIPELINING
223	16.125844	10.251.137.55	183.47.101.192	SMTP	66 C:	AUTH login
224	16.143559	183.47.101.192	10.251.137.55	SMTP	72 S:	334 VXNlcm5hbWU6
225	16.144064	10.251.137.55	183.47.101.192	SMTP	81 C:	User: User:
226	16.156517	183.47.101.192	10.251.137.55	SMTP	72 S:	334 UGFzc3dvcmQ6
227	16.157060	10.251.137.55	183.47.101.192	SMTP	81 C:	Pass: Pass:
229	16.341503	183.47.101.192	10.251.137.55	SMTP	85 S:	235 Authentication successful
230	16.342386	10.251.137.55	183.47.101.192	SMTP	84 C:	mail from:<691745382@qq.com>
232	16.393770	183.47.101.192	10.251.137.55	SMTP	62 S:	250 OK
233	16.394357	10.251.137.55	183.47.101.192	SMTP	82 C:	rcpt to:<691745382@qq.com>
235	16.463060	183.47.101.192	10.251.137.55	SMTP	62 S:	250 OK
236	16.463622	10.251.137.55	183.47.101.192	SMTP	60 C:	data
238	16.477793	183.47.101.192	10.251.137.55	SMTP	92 S:	354 End data with <CR><LF>.<CR><LF>.
239	16.488596	10.251.137.55	183.47.101.192	SMTP	1078 C:	DATA fragment, 1024 bytes
240	16.488614	10.251.137.55	183.47.101.192	SMTP	1478 C:	DATA fragment, 1424 bytes
241	16.488614	10.251.137.55	183.47.101.192	SMTP	678 C:	DATA fragment, 624 bytes
242	16.488627	10.251.137.55	183.47.101.192	SMTP	1478 C:	DATA fragment, 1424 bytes
< Frame 219: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface \Device\NPF_{4641C868-97C3-4BF5-BD1F-130641760} >						
> Ethernet II, Src: HuaweiTe_77:54:52 (00:2e:c7:77:54:52), Dst: EliteGro_20:ba:df (1c:69:7a:20:ba:df)						
> Internet Protocol Version 4, Src: 183.47.101.192, Dst: 10.251.137.55						
> Transmission Control Protocol, Src Port: 25, Dst Port: 51689, Seq: 1, Ack: 1, Len: 65						
> Simple Mail Transfer Protocol						
<						
0000	1c 69 7a 20 ba df 00 2e c7 77 54 52 08 00 45 04	.izwTR..E-				
0010	00 69 63 0f 40 00 30 06 36 5a b7 2f 65 c0 0a fb	.ic.@.0. 6Z-/e...				
0020	89 37 00 19 c9 e9 10 be 69 c2 25 11 a2 06 50 18	.7.....i.%...P-				
0030	00 e5 48 49 00 00 32 32 30 20 6e 65 77 78 6d 65	..HI..22 0 newxme				
0040	73 6d 74 70 6c 6f 67 69 63 73 76 72 73 7a 62 31	smtplogi csvrszb1				
0050	2d 30 2e 71 71 2e 63 6f 6d 20 58 4d 61 69 6c 20	-0.qq.co m XMail				
0060	45 73 6d 74 70 20 51 51 20 4d 61 69 6c 20 53 65	Esmtpl QQ Mail Se				
0070	72 76 65 72 2e 0d 0a	rver..				

2. 邮件接收客户端实验结果及分析

命令：`./send -s "Mail subject" -m message.txt -a "attachment.zip" example@example.org`
`./recv`

主题为 Mail subject，message.txt 是邮件正文的文件路径，邮件附件为 attachment.zip

各文件内容与上述一致。

结果如下：


```

[05/22/23]seed@VM:~/maillab-master$ ./recv
+OK XMail POP3 Server v1.0 Service Ready(XMail v1.0)
user 691745382@qq.com
+OK
pass 
+OK
stat
+OK 4 30450
list
+OK
1 4171
2 3907
3 8528
4 13844
.
retr 1
+OK 4171
quit
X-QQ-XMAILINFO: N1woPqWMJ6QLiEuV7ztNHCjNzbxiKVGnKhsm4YrCDUOP5r3CIGX+bftPX4t/En
JWjEDWz48Zh5QTYGVuE+Njp8h7Y7jlfkNdioW57vhYrmFh80dxIcJeGs71uBjWkqSNFSgVpeDk+aX
Bx26PYdWyeIh9+j3T4jsA4HKLcwIv+Ew+BxGN+KBdaiKRo2mZhYMSimTgX3VdkYTnw/eXj0GdIjEC
1K1TtrZutlx6ZlRHo9zduH5iNoeZKxLfaGj0RAMpkbpD3PStrUg80T2pXWqBCixL+cVfDtSFIEZVX
RNJZ2lq03pcrPLlXrnoUB+BDNf0ItlmASWLhf0RdeT6GavBC90YNVw+YK/LEXpDwcyhvhrx9cUgxZ
ZMDbadhdbKibJ0I3Ssz6LV27uhea4z4ZMnAEnP1lgS5r0mG0psPvIzsTVhrujuPUWch13m9cba165
5SrYw1td9lXHjAvVablPnKp6uB/dKhCprwMZdRoPuSEpIG+v0buHY6lEb5DntjVR9PA+bwTNn2osd
bm10YdXjn/UnpPQK1n1UjiiKgkY06zARJu+JgPlZ+212B1lMeF0jlve+MS95keQ9Foiff07z8qKV
qYjX0wlzDGLuEGRfQ4KUHyfFC25s9LgEbZDifgVcjFxpCiFPy9exPQG+LC5X+J9PkIrYQvbwgW0Y
aryPycVA00JE1aBv7gyyCwFATDAhJwCd/RnmFf9bIPFJmKqd0pGgwL7mIX2c/rLTt2EvaODPYaidr
CJer/C93EBzdozqGVypZWYm7qCAGPF2rTgmCJ9YqoSPTjByEsm7/B0DrtrwN2efex+8vkdHEELxg/
oh8uA0o0K0qHo9SLERWqUKjy0zhWhjb2dkYib0G6MiySWaHrnaMfl05giV01ldMeIfkXiUNJZ8M09
+DL32RsuqBTdtgozb8Y5WtefY9DKuynQJ4FMYFNXs1JdVhC4r+0oGe4Rzs1RBtMf0luXoQB/JLLVf
A07p41fw==
X-QQ-INNER-SUBJECT-ID: 19771

X-QQ-INNER-SUBJECT-ID: 19771
X-QQ-INNER-ID: 19771
From: "=?utf-8?B?UVHpn7PkuZDpobnnm67nu4Q=?=" <music@tencent.com>
To: "=?utf-8?B?NjKxNzQ1Mzgy=?" <691745382@qq.com>
Subject: =?utf-8?B?5oGt5Zac5L2g77yM6LGq5Y2057u/6ZK757ut6LS5?=?
=?utf-8?B?5oiQ5Yqf=?
Mime-Version: 1.0
Content-Type: multipart/alternative;
    boundary="-----_NextPart_6463A153_449C65A0_72ABD856"
Content-Transfer-Encoding: 8Bit
Date: Tue, 16 May 2023 23:29:23 +0800
X-Priority: 3
Message-ID: <tencent_75EAEEBFBC5D38F3C3A74498C055AE84D10A@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: sysmailb50t1684250963t9ly35cdf

This is a multi-part message in MIME format.

-----_NextPart_6463A153_449C65A0_72ABD856
Content-Type: text/plain;
    charset="utf-8"
Content-Transfer-Encoding: base64

5oGt5Zac5L2g77yM6LGq5Y2057u/6ZK757ut6LS55oiQ5YqfKDIwMjMtMDUtMTYmbmJzcDsy
Mzoy0ToyMikNCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
NS0xNiZuYnNwOzIzOjI5OjIy57ut6LS5MeS4quaci0ixquWNjue7v+mSu++8j0acieaVi0ac
n+iHszIwMjMtMDYtMTcmbmJzcDsxNzoiNjowM+0Agg0KICAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICDnmb7pobnkuJPkuqvnibnmnYPvvIzov5jmnInlpJrIsJHmnKrkvb/nlKjv
vJ/ku4rml6XmjqqojZDkvJrIkZjmm7LlupPjgIHkuIvovb3nibnmnYPjgIHkuKrmgKfkuLvp
opjnrYnotoXlpJrnibnmnYPjgILLhbPms6giUVHpn7PkuZBWSVAi5b6u5L+h5YWs5LyX5Y+3

```

```
77yM5pi05pif5ZGo6L6544CB5LiT5bGe5oq15omj5Yi444CB5ryU5ZSx5Lya6Zeo56Wo44CB
55S15b2x56Wo562J56aP5Yip77yM5oqi5YWI6aKG77yBDQogICAgICAgICAgICAg6L+b5YWL
5a6Y572R
```

```
-----=_NextPart_6463A153_449C65A0_72ABD856
Content-Type: text/html;
    charset="utf-8"
Content-Transfer-Encoding: base64
```

```
IDxkaXYgY2xhc3M9Im9wZW5fZW1haWwgY2x1YXJmaXgiIHN0eWxlPSJtYXJnaW4tbGVmdDog
0HB40yBtYXJnaW4tdG9wOiaA4cHg7IGlhcmdpbilib3R0b206IDhweDsgbWfY2ZluLXJpZ2h0
0iaA4cHg7Ij4NCiAgICAgICAgPGRpdjBjbGFzc0iaWlnX2xlZnQiPg0KICAgICAgICAgICAg
PGEgaHJlZj0iaHR0cDovL20ucXpvcUuY29tL2w/Zz0yODI3Ij4NCiAgICAgICAgICAgICAg
ICA8aWlnIHNYZ0iaHR0cDovL3kuZ3RpbWcuY24vbXVzaWMvcGhvdG9fbmV3L1QwMTFNMDAw
MDAxR3VhNGQyUTZuTUkuanBnIiBoZWlnaHQ9IjkwIkw0KICAgICAgICAgICAgICAgICAgICAg
aWR0aD0iMTI1IiAvPjwvYT48L2Rpdj4NCiAgICAgICAgPGRpdjBjbGFzc0id29yZF9yaWdo
dCI+DQoNCiAgICAgICAgICAgICAgIDxoNT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgIC
sarljY7nu7/pskrvnu63otLnmiJDlip88c3Bhb40MjAyMy0wNS0xNiZuYnNwOzIz0jI50jIy
KTWvc3Bhb48L2g1Pg0KQogICAgICAgICAgICA8aDY+DQogICAgICAgICAgICAgICAgICAg5L2g
5LQ0MjAyMy0wNS0xNiZuYnNwOzIz0jI50jIy57ut6LS5MeS4quaci0ixquWNjue7v+mSu++8
j0acieaVi0acn+iHszIwMjMtMDYtMTcmbmJzcDsxNzo1NjowM+0AgjwvaDY+DQogICAgICAg
ICAgICA8cD4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAg
pJrlsJHmnKrkvb/nlKjvvJ/ku4rml6XmjQojZDkvJrlkZjmm7LlupPjgIHkuIvovb3nibnm
nYPjgIHkuKrmgKfkuLvpopjnrYnotoXlpJrnibnmnYPjgILlhbPms6giUVHpn7PkuZBWSVAi
5b6u5L+h5YWs5LyX5Y+377yM5pi05pif5ZGo6L6544CB5LiT5bGe5oq15omj5Yi444CB5ryU
5ZSx5Lya6Zeo56Wo44CB55S15b2x56Wo562J56aP5Yip77yM5oqi5YWI6aKG77yBPC9wPg0K
ICAgICAgICAgICAgPGEgY2xhc3M9ImVtYWlsX2J0biIgdGFyZ2V0PSJfYmxhbmsiIGhyZWY9
Imh0dHA6Ly9tLnF6b25lLnNvbS8iPui/m+WFpeWum0e9kTwvYT48L2Rpdj4NCiAgICA8L2Rp
dj4NCg==
```

```
-----=_NextPart_6463A153_449C65A0_72ABD856--
```

```
.
[05/22/23] seed@VM:~/maillab-master$ █
```

如上图所示，总邮件个数及大小为：

```
stat
+OK 4 30450
```

每封邮件的编号及大小为

```
list
+OK
1 4171
2 3907
3 8528
4 13844
```

第一封邮件的内容如图中其余部分所示。

收到的邮件内容如发送结果除所示，不再赘述。

Wireshark 抓包结果如下，均符合预期：

No.	Time	Source	Destination	Protocol	Length	Info
305	20.554707	183.47.101.192	10.251.137.55	POP	108	S: +OK XMail POP3 Server v1.0 Service Ready(XMail v1
306	20.556592	10.251.137.55	183.47.101.192	POP	77	C: user 691745382@qq.com
308	20.571005	183.47.101.192	10.251.137.55	POP	60	S: +OK
309	20.571509	10.251.137.55	183.47.101.192	POP	77	C: pass sghqmxmttvmeic
315	20.774370	183.47.101.192	10.251.137.55	POP	60	S: +OK
316	20.776180	10.251.137.55	183.47.101.192	POP	60	C: stat
318	20.794855	183.47.101.192	10.251.137.55	POP	69	S: +OK 27 335669
319	20.795355	10.251.137.55	183.47.101.192	POP	60	C: list
321	20.836200	183.47.101.192	10.251.137.55	POP	112	S: +OK
322	20.836701	10.251.137.55	183.47.101.192	POP	62	C: retr 1

> Frame 305: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface \Device\NPF_{4641C868-97C3-4BF5-8D1F-130641766}
> Ethernet II, Src: HuaweiTe_77:54:52 (00:2e:c7:77:54:52), Dst: EliteGro_20:ba:df (1c:69:7a:20:ba:df)
> Internet Protocol Version 4, Src: 183.47.101.192, Dst: 10.251.137.55
> Transmission Control Protocol, Src Port: 110, Dst Port: 51691, Seq: 1, Ack: 1, Len: 54
> Post Office Protocol

0000	1c 69 7a 20 ba df 00 2e c7 77 54 52 08 00 45 04	·iz ··· ·wTR··E·
0010	00 5e 71 69 40 00 30 06 28 0b b7 2f 65 c0 0a fb	·^qi@·0· (·/e··
0020	89 37 00 6e c9 eb 58 3e e7 84 9a c4 6c bf 50 18	·7·n·X> ····1·P·
0030	01 f6 e3 a8 00 00 2b 4f 4b 20 58 4d 61 69 6c 20	·····+O K XMail
0040	50 4f 50 33 20 53 65 72 76 65 72 20 76 31 2e 30	POP3 Ser ver v1.0
0050	20 53 65 72 76 69 63 65 20 52 65 61 64 79 28 58	Service Ready(X
0060	4d 61 69 6c 20 76 31 2e 30 29 0d 0a	Mail v1.0)··

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)

在实现邮件正文内容填写（msg 表示的是存储邮件正文内容的文件地址）时，我出现了错误。（下述代码块）

```
if(file_mail_body) {
    // msg = path to the file containing mail body
    int mail_size = fread(buf, 1, MAX_SIZE, file_mail_body);
    buf[mail_size] = '\0';
    strcat(buf, "\r\n");
    send(s_fd, buf, strlen(buf), 0);
    printf("%s", buf);
    fclose(file_mail_body);
}
```

起初我直接将 fread 读到的内容复制进 buf 中，在后面添加换行符后就发送出去了。但这样实现会导致最终发送出去的内容中存在上一条指令的后半部分。经研究我发现是由于 fread 读取到的内容并没有结束符，因此 buf 中的其余内容也被一并输出了。于是我通过 fread 得到的文件大小，在 buf 的相应位置添加了结束符，然后再添加换行符，将信息发送出去，解决了这一错误。

在最后的分析时，我同样遇到了问题。起初我将代码打包到实验室的 A100 服务器上最终结果的验证，但我发现得到的输出会产生错位，如下图所示：

```

(base) zq@a100-40g:/data/zq/maillab-master$ make
gcc -c base64_utils.c -o base64_utils.o
gcc -c cencode.c -o cencode.o
gcc -c cdecode.c -o cdecode.o
gcc send.c base64_utils.o cencode.o cdecode.o -o send
gcc recv.c -o recv
(base) zq@a100-40g:/data/zq/maillab-master$ ./send -s "Mail subject" -m message.txt -a "attachment.zip" 691745382@qq.com
2EHLO qq.com
20 newxmesmtplogicsvrszb6-0.qq.com XMail Esmtp QQ Mail Server.
AUTH login
250-newxmesmtplogicsvrszb6-0.qq.com
250-PIPELINING
250-SIZE 73400320
250-STARTTLS
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2
250-AUTH=LOGIN
250-MAILCOMPRESS
250 8BITMIME

334 VXNlcm5hbWU6

334 UGFzc3dvcmQ6
mail from:<691745382@qq.com>
235 Authentication successful
rcpt to:<691745382@qq.com>
250 OK
data
250 OK
From:691745382@qq.com
To:691745382@qq.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiopasdfghjklzxcvbnm
Subject:Mail subject

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/plain

Computer networking is so much fun!

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

.
354 End data with <CR><LF>.<CR><LF>.
quit
250 OK: queued as.
(base) zq@a100-40g:/data/zq/maillab-master$

```

经分析，我认为这是因为代码中的输出是使用 `printf` 实现的，`printf` 并不是线程安全的，而该服务器可能在运行时实现了多线程，所以会导致输出错误。后来我将代码拷到 t2 教室的虚拟机上进行结果的验证，最终得到了正确的输出。

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。）

本学期计算机网络的三种类型的实验起到了很好的平衡效果。配置验证实验较为简单轻松，穿插在协议栈系列实验和 socket 编程实验中，降低了完成实验的压力，但同时

配置验证实验也能很好地让我们了解到数据报的具体传输过程，是一个很不错的实验类型。

协议栈类型的实验相比于配置验证实验要困难很多，尤其是实验过程中是按照层次完成的，有出现过当前层的测试不通过，但最终发现是前一层的问题。对此，我认为可以适当地将评分代码进行完善，以便能查找出当前层可能出现的问题，而不至于累积到后续的实验中再发现，不然会给 debug 造成较大的困难。前几层的实验中，指导书上的提示都很详细，虽然刚开始上手比较困难，但跟着指导书一步一步做还是能够顺利完成的，但后面几层的提示可能就没有那么详细了。

Socket 编程的实验相比于配置验证实验也要更困难，尤其是指导书上的提示比较分散，个人建议可以把附录中的内容放到相应的实验部分，不然一开始看的时候很有可能一头雾水，不知道从何下手。不过完成实验的过程还是很有趣的，尤其是能够通过代码直接给邮箱发邮件，完成实验后感觉很有成就感。

总体而言，我觉得这学期的计算机网络实验安排的非常合理，难度适中，即帮助我们熟悉了理论课上所学的知识，也没有给我们造成太大的实验负担。