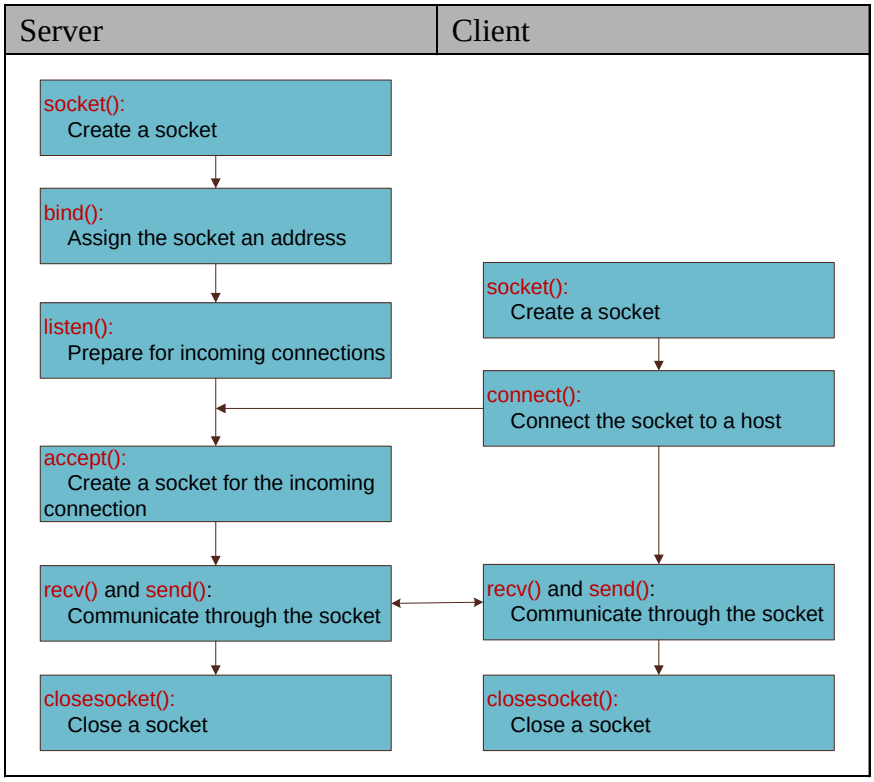
Introduction to Computer Networks
Final Project Report

108062111 劉柔恣

- Details of your implementation, including server-side and client-side.

在 TCP 協定中，需要先建立起兩端的連線，接著才能開始進行資料傳輸。



所以，在程式中，必須先根據 TCP flow chart，實作出 server 和 client 的連結，接著再實作出不同功能來處理資料內容，如印出留言板內容、以及接收新的留言等等。

以下逐項講解程式碼細節。

- server 端

```
serverSocket = socket(PF_INET, SOCK_STREAM, 0); // protocol family, T
if(serverSocket < 0){
    fprintf(stderr, "Error creating socket : %s\n", strerror(errno));
    exit(0);
}
```

首先，先建立起一個 serverSocket，以供監聽用。

此處 socket() 內使用的 pf family 是 AF_INET，代表使用 IPv4 來表示網路位址；type 是 SOCK_STREAM，因為 TCP 是 byte-oriented；protocol 則填入 0，由系統來決定。

```
bzero(&serverAddress, server_addr_length);
serverAddress.sin_family = AF_INET; // address family: Internet family, IPv4
serverAddress.sin_port = htons(ServerPortNumber); // "Host byte order" to "Network b
serverAddress.sin_addr.s_addr = INADDR_ANY;

if(bind(serverSocket,(struct sockaddr *) &serverAddress, server_addr_length) == -1){
    fprintf(stderr, "Error binding : %s\n", strerror(errno));
    close(serverSocket);
    exit(0);
}
```

接著，設定好 server 的 address 內容後，將該 address 綁在 serverSocket 上。

```
if(listen(serverSocket, 3) == -1){
    fprintf(stderr, "Error listening : %s\n", strerror(errno));
    close(serverSocket);
    exit(0);
}
```

讓 serverSocket 進入監聽的狀態。

```
printf("Waiting...\n");
if((clientSocket = accept(serverSocket, (struct sockaddr *)&clientAddress, &client_addr_length)) == -1){
    // accept: return new socket descriptor
    // serverSocket is still listening
    printf("accept failed\n");
    close(serverSocket);
    exit(0);
}
printf("Client connect successfully\n");
```

當 serverSocket 監聽到 client 發出 connect 的請求後，使用 accept() 來處理。
accept() 會回傳一個新的 socket descriptor，接著就能透過該 socket（clientSocket）進行 server 和 client 之間的資料傳輸。
原本的 serverSocket 則繼續保持監聽。

接著處理資料的部分：

```
#define MAXNUM 10
#define MAXLEN 30
#define MAXBUF 500

int messageNumber = 0;
char messageList[MAXNUM][MAXLEN];
char send_buf[MAXBUF];
char recv_buf[MAXBUF];
    char *menu = "\
\n\n--- Menu ---\n\
1. Read all existing massages.\n\
2. Write a new message.\n\
Please enter your operation : \0";

    char *show = "All existing messages :\n\0";
    char *input = "Please enter your message : \0";
    char *isFull = "The message board is full. You cannot enter new message.\0";
```

首先，我使用 messageNumber 來記錄現在的留言板有幾則留言、使用 messageList 來記錄留言的內容。
buffer 的大小是 MAXBUF，參考了助教的 sample code，設為 500；而留言的數

量上限是 MAXNUM、每則留言的字數上限是 MAXLEN，考慮到 buffer 的大小，我將 MAXNUM 設為 10、將 MAXLEN 設為 30。兩者相乘為 300，留下 200 的空間給 menu、show、input、isFull 等訊息，以及結束符一類的特殊字元使用。

```
// Send menu to client
send_buf[0] = '\0';
strcat(send_buf, menu);
bytesSend = send(clientSocket, send_buf, sizeof(send_buf), 0);
if(bytesSend < 0) printf("Error sending packet\n");

while(1){
    bytesRecv = recv(clientSocket, recv_buf, sizeof(recv_buf), 0);
    if(bytesRecv < 0) printf("Error receiving packet\n");
```

server 會先寄送 menu 給 client，接著在 while 迴圈內，使用 recv()，不停接收 client 寄送的資料，再根據資料中的不同指令，進行相對應的工作。

```
if(!strncmp(recv_buf, "1", 1)){
    send_buf[0] = '\0';
    strcat(send_buf, show);
    for(i = 0; i < messageNumber; i++){
        strcat(send_buf, messageList[i]);
        strcat(send_buf, "\n\0");
    }
    strcat(send_buf, menu);
    bytesSend = send(clientSocket, send_buf, sizeof(send_buf), 0);
    if(bytesSend < 0) printf("Error sending packet\n");
}
```

當指令為 1 時，會印出留言板中的所有留言。
此處我使用了 strcat() 將 show、所有留言、menu 串起來放在 send_buf 中，接著使用 send() 將資料送出去。

```
else if(!strncmp(recv_buf, "2", 1)){
    if(messageNumber >= MAXNUM){
        send_buf[0] = '\0';
        strcat(send_buf, isFull);
        strcat(send_buf, menu);
        bytesSend = send(clientSocket, send_buf, sizeof(send_buf), 0);
        if(bytesSend < 0) printf("Error sending packet\n");
    }
```

當指令為 2 時，代表要輸入新的留言。
此處會先檢查當前的留言數量是否已達上限，如果已經到達上限，則會顯示留言板已滿的訊息，以及 menu 的內容。

```
else {
    send_buf[0] = '\0';
    strcat(send_buf, input);
    bytesSend = send(clientSocket, send_buf, sizeof(send_buf), 0);
    if(bytesSend < 0) printf("Error sending packet\n");

    bytesRecv = recv(clientSocket, recv_buf, sizeof(recv_buf), 0);
    if(bytesRecv < 0) printf("Error receiving packet\n");
    printf("%s!\n", recv_buf);

    strcpy(messageList[messageNumber++], recv_buf);
    messageList[messageNumber - 1][MAXLEN - 1] = '\0';

    send_buf[0] = '\0';
    strcat(send_buf, menu);
    bytesSend = send(clientSocket, send_buf, sizeof(send_buf), 0);
    if(bytesSend < 0) printf("Error sending packet\n");
}
```

如果留言板還沒滿，則會詢問 client 要輸入什麼內容，接著等待 client 的回應。
收到 client 的資料後，則使用 strcpy() 將內容複製到 messageList 中。
為了防止溢位，將 messageList[messageNumber - 1][MAXLEN - 1]設為結束符。
接著將 menu 寄給 client，等待 client 的下一個指令。

```
else{
    bytesSend = send(clientSocket, menu, strlen(menu), 0);
    if(bytesSend < 0) printf("Error sending packet\n");
}
```

如果 client 輸入了 1、2 以外的指令，則不做任何事，直接將 menu 寄給 client，等待 client 的下一個指令。

- client 端

```
// Create socket
serverSocket = socket(PF_INET, SOCK_STREAM, 0);
if(serverSocket < 0){
    printf("Error creating socket\n");
    exit(0);
}
```

一樣先建立起 socket，此處的設定同 server。

```
// Set the server information
bzero(&serverAddress, server_addr_length);
serverAddress.sin_family = AF_INET;
serverAddress.sin_port = htons(ServerPortNumber);
serverAddress.sin_addr.s_addr = inet_addr(ServerIP);
```

接著，設定好 server 的 address 內容。

```c
// Connect to server
if(connect(serverSocket, (struct sockaddr *)&serverAddress, server_addr_length) == -1){
    printf("connect failed\n");
    close(serverSocket);
    exit(0);
}
```

然後，使用 connect() 向 server 發出連線的請求。

```c
while(1){
    bytesRecv = recv(serverSocket, recv_buf, sizeof(recv_buf), 0);
    if(bytesRecv < 0) printf("Error recving packet\n");
    printf("%s\n", recv_buf);

    fflush(stdin);
    scanf(" %[^\n]", send_buf); // receive any character except '\n'

    bytesSend = send(serverSocket, send_buf, sizeof(send_buf), 0);
    if(bytesSend < 0) printf("Error sending packet\n");

}
```

當連線建立後，就會開始進行資料的傳輸。
此處會將收到的資料 recv_buf 印出來，接著傳送使用者輸入的內容給 server。

- Step-by-step screenshots and explanations of the execution of each function.

1.開啟 server，等待 client 連線。

```
diasi@diasi-ubuntu:~/Desktop/I2CN$ ls
cli  client.c  ser  server.c
diasi@diasi-ubuntu:~/Desktop/I2CN$ ./ser 5555
Waiting...
```

2.開啟 client，並連線至 server。

```
diasi@diasi-ubuntu:~$ cd Desktop/I2CN/
diasi@diasi-ubuntu:~/Desktop/I2CN$ ls
cli  client.c  ser  server.c
diasi@diasi-ubuntu:~/Desktop/I2CN$ ./cli 127.0.0.1 5555


--- Menu ---
1. Read all existing massages.
2. Write a new message.
Please enter your operation :
```

3.輸入指令 2，並輸入新的留言。

```
--- Menu ---
1. Read all existing massages.
2. Write a new message.
Please enter your operation :
2
Please enter your message :
Hello World!
```

4.輸入指令 1，查看所有留言。

```
--- Menu ---
1. Read all existing massages.
2. Write a new message.
Please enter your operation :
1
All existing messages :
Hello World!
```

5.當留言板的留言數量達到上限（此處為 10 則留言），會顯示留言板滿了的訊息，並且無法再繼續輸入新的留言。

```
--- Menu ---
1. Read all existing massages.
2. Write a new message.
Please enter your operation :
1
All existing messages :
Hello World!
2
3
4
5
6
7
8
9
10


--- Menu ---
1. Read all existing massages.
2. Write a new message.
Please enter your operation :
2
The message board is full. You cannot enter new message.
```

- Descriptions of difficulties you encountered and your solutions.

一開始，我在 windows 中編譯、執行助教給的 sample code，但因為不了解 socket 在 windows 和 linux 環境中的差異，所以遇到許多錯誤，花了不少時間 google 找資料、除錯，後來再看助教給的 tutorial，裡面其實都有提到兩者之間的差異：

	linux	windows
header	#include <sys/socket.h> #include <netinet/in.h>	#include <winsock2.h>
compile 指令	gcc -o ser server.c	gcc -o ser server.c -lws2_32
初始化 windows sockets dll	無	WSAData wsaData; WORD version=MAKEWORD(2,2); WSAStartup(version, &wsaData);
終止 dll	無	WSACleanup();

其中，我卡最久的是：在執行 server 的程式時，一直出現「Error creating socket : No error」。上方表格的內容我都已經改了，但仍然沒辦法順利執行。

也試過在程式碼最前面加入其他 header：

```
#include <Windows.h>
#include <ws2tcpip.h>
```

以及在程式碼中直接加入 library 的連結：

```
#pragma comment(lib, "Ws2_32.lib")
```

但都還是沒有用。

最後，因為找不到問題的所在，所以我轉而去使用虛擬機以及安裝 ubuntu 了。雖然暫時還沒辦法成功在 windows 跑 socket，但是設置好虛擬機和 ubuntu 的環境也是我在這次 final project 中的收穫之一。

另外，在撰寫印出留言板所有留言的功能時，我有考慮過每一條留言都呼叫一次 send()，這樣一來，留言的長度以及數量比較不會受到 buffer 的限制，但實際執行時卻發現：明明底下沒有放 recv()，但是 server 一定要等到收到 client 寄來的資料後，才會進行下一次的 send()。

起初不知其所以然，只是單純修改成把所有留言都塞到 send_buf 後再一次寄出。後來領悟到原來這就是 echo server、echo client 的機制啊，訊息之間必須要一來一往，不能單方面送出很多筆訊息。