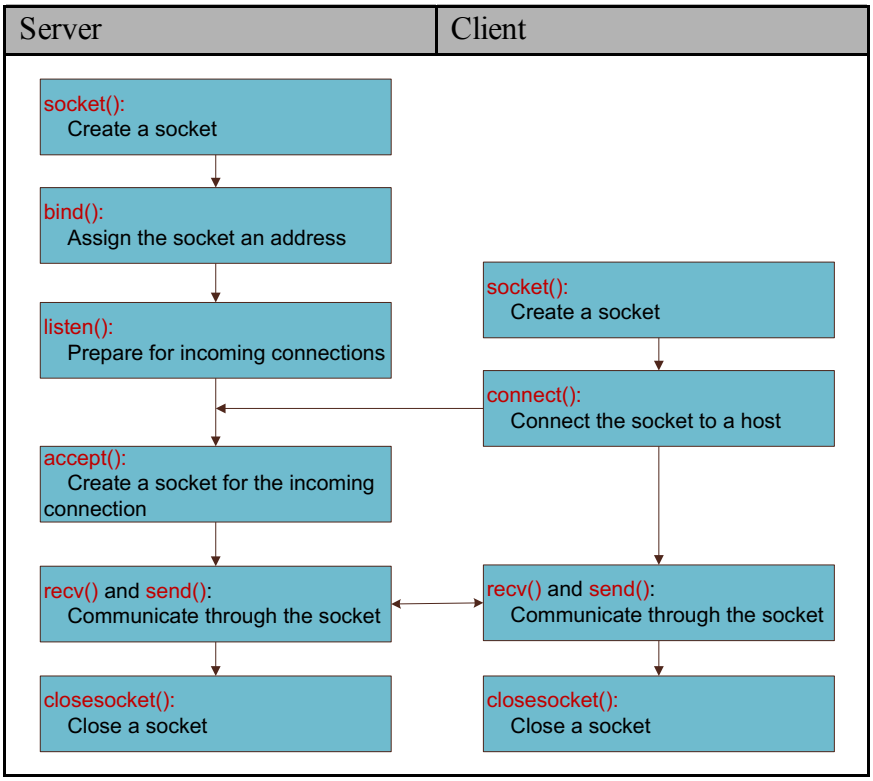
Computer Networks socket practice report

107070008 陳睿芊

Part I. implementation

首先，參考了 socket tutorial 裡的設置去建立 server 和 client 之間的連結。



而講義上所說要以 Tcp 為設計標準，於是參考這張圖的架構以及 ppt 裡提供的一些 socket function。

將 server 和 client 連結起來後，下面實作選單，選單主要靠 send_buf 和 recv_buf 傳送，卻傳送無誤之後就完成了。

Part.II step-by-step code explanation

1. 首先將會用到的 lib 都 include 進來。這邊我將 Port 寫死為 8000。

   (這裏 client 和 server 都相同)

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

#define PORT 8000
```

2. 然後是宣告要用到的變數，右邊是 server，左邊是 client。在 socket tutorial 中有提到的 AF_INET 是 address family，htons 是設定他的 port，而 client 這裡接的 ip address 是 127.0.0.1，代表連上本機自己。

```c
int main(void)
{
    struct sockaddr_in ser_addr, cli_addr;
    int ser_socket, cli_socket;
    int ser_addr_len = sizeof(ser_addr);
    int cli_addr_len = sizeof(cli_addr);

    bzero(&ser_addr, ser_addr_len);
    ser_addr.sin_family = AF_INET;
    ser_addr.sin_port = htons(PORT);
    ser_addr.sin_addr.s_addr = INADDR_ANY;
```

```c
int main(void)
{
    struct sockaddr_in ser_addr;
    int ser_socket;
    int ser_addr_len = sizeof(ser_addr);

    bzero(&ser_addr, ser_addr_len);
    ser_addr.sin_family = AF_INET;
    ser_addr.sin_port = htons(PORT);
    ser_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

參考這一頁 ppt：

Data structure of address

Structure	Usage

struct sockaddr_in {
short sin_family;
unsigned short sin_port;
struct in_addr sin_addr;
char sin_zero[8];
};
	sin_family = AF_INET; (Address Family)
	sin_port = htons(80);
	sin_addr=inet_addr("127.0.0.1"); (for client)
	sin_addr.s_addr = INADDR_ANY; (for server)

The **htons** function converts a **u_short** from host to TCP/IP network byte order (which is big-endian).

3. 接著按照 Part 1 圖的架構開始實作，首先在 server 和 client 都先 create a socket，這邊因為題目規定要使用 TCP，所以在 socket function 的 type 中的參數必須是 socket_stream。（這個 function 的設定在 server 和 socket 皆相同）而下面依據助教所設的 debug 訊息，有助於在編程時除錯，因爲 socket 回傳–1 時表示錯誤，所以 if(ser_socket < 0)。

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

4. 接著 server 實行架構中的 bind 和 listen，function 依照 ppt 給定的說明將參數設定好。也依樣在下方設定 debug 訊息方便除錯。

- **int bind(int sockdes, const struct sockaddr *addr, socklen_t addrlen)**
  bind()是把設定的address綁在Socket身上
  參數：
  sockdes：指定好通訊協定的socket
  addr：指定本地端位址，資料格式為sockaddr
  addrlen：addr之資料長度(單位byte)
  回傳值：-1表錯誤，否則為0

- **int listen(int sockdes, int backlog)**

  等待請求
  參數：
  sockdes：設定好bind(),並且尚未連線的socket
  backlog：等待Server接受連線前，同時最大連線數
  回傳值：-1表錯誤，否則為0

```
if(bind(ser_socket,(struct sockaddr *) &ser_addr, ser_addr_len) == -1){
    printf("Error binding\n");
    close(ser_socket);
    exit(0);
}

if(listen(ser_socket, 1) == -1){
    printf("Error listening\n");
    close(ser_socket);
    exit(0);
}
```

5.仍是照著 tcp socket 架構，先設定 client 的 connect function 使其與 server 連結，一樣參照 ppt 裡的 function 設定參數。

- **int connect(int sockdes, const struct sockaddr *addr, socklen_t addrlen)**

  建立連線
  設定方式請參照bind()函式
  回傳值：-1表錯誤，否則回傳0

```
if(connect(ser_socket, (struct sockaddr *)&ser_addr, ser_addr_len) == -1){
    printf("connect failed\n");
    close(ser_socket);
    exit(0);
}
```

6. 這裏 server 接收 client 的的連線，成功 print connect successfully，錯誤則 print accept fail。

- **int accept(int sockdes, struct sockaddr *addr, socklen_t *addrlen)**

  接受請求
  參數：
  sockdes：一個設定為listen狀態的socket
  addr：Client端位址資訊
  addrlen：addr長度
  回傳值：-1表示錯誤，否則傳回另一個包含Client端資訊的新socket descriptor，作為傳送資料用

```
printf("Waiting to the client!\n");
if((cli_socket = accept(ser_socket, (struct sockaddr *)&cli_addr, (socklen_t *)&cli_addr_len)) == -1){
    printf("accept failed\n");
    close(ser_socket);
    exit(0);
}
printf("Client connect successfully\n");
```

這邊一樣餐考 ppt 並於下面加入 debug 訊息。

7. 以上實施成功則成功連結 client 和 server 端。接下來 implement menu 和 sever&client 之間的 recv()和 send()。先將要用的變數宣告完成，這裏包括 menu 和儲存所有訊息的陣列，以便日後 choose 1 時可以一起 print 出。

（以下左邊為 sever 端的設定，右邊為 client 端）

```
    int bytesRecv, bytesSend;
    char send_buf[500], recv_buf[500], messagebox[500];
    messagebox[0] = '\0';
    char *menu = "\
\n\n------------------Menu------------------\n\
1. Read all the messages.\n\
2. Write new messages.\n\
Please enter 1 or 2 : \0";

    char *messagelist = "\
\n\n----------------messages----------------\n\
All Messages:\n\
First line here.\0";

    char *newitem = "\
\n\n----------------------------------------\n\
Type anything you want:\0";

    char *errorlist = "\
\n\n--------------Warning!----------------\n\
you can only enter operation 1 or 2 !\0";

    char *changeline = "\n";
```

```
int bytesSend, bytesRecv;
char send_buf[500], recv_buf[500];
```

8. 這裏先說明 server 機制，首先先將 menu 傳給 client 讓 client 印出。

send()和 recv() function 的設定機制一樣，參考 ppt。

- **ssize_t recv(int sockdes, void *buf, size_t len, int flags)**
  參數：
  sockdes：一個建立連線成功的socket
  buf：呼叫recv，用來儲存收到資料的暫存器
  len：buf的長度(byte)
  flags：選擇工作模式，一般填入0
  回傳值：-1表錯誤，否則傳回接受到資料的長度(byte)

- **ssize_t send(int sockdes, const void *buf, size_t len, int flags)**
  參數：
  sockdes：一個建立連線成功的socket
  buf：用來儲存將送出資料的暫存器
  len：buf的長度(byte)
  flags：選擇工作模式，一般填入0
  回傳值：-1表錯誤，否則傳回送出資料的長度(byte)

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

9. 接著等接收 client user 的 I/O，這個過程要一直重複，所以寫在 while 裡。將收的資料 recv_buf 做判定，若字串是 1 則進行第一項，print all the message，將 messagelist 和所有之前儲存的 message 再加上 menu(這裏是為了進行下一格輸入輪迴）一起剪到 send_buf 之中，傳給 client 印出。

若字串是 2 則進行第二項，讓使用者輸入新的 message。先將 newitem 的字樣寫到 send_buf 中傳給 client 印出之後，等待 client 那邊使用者的 I/O 回傳。完了之後再 send 一次 menu 字串給 client 以便進行下一次選項輸入輪迴。

（下圖）

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

    printf("%s\n", recv_buf);

    if(!strncmp(recv_buf, "1", 1)){
        send_buf[0] = '\0';
        strcat(send_buf, messagelist);
        if(messagebox[0]!='\0') strcat(send_buf, messagebox);
        strcat(send_buf, menu);
        bytesSend = send(cli_socket, send_buf, sizeof(send_buf), 0);
        if(bytesSend < 0) printf("Error sending packet\n");
    }
    else if(!strncmp(recv_buf, "2", 1)){
        send_buf[0] = '\0';
        strcat(send_buf, newitem);
        bytesSend = send(cli_socket, send_buf, sizeof(send_buf), 0);
        if(bytesSend < 0) printf("Error sending packet\n");

        bytesRecv = recv(cli_socket, recv_buf, sizeof(recv_buf), 0);
        if(bytesRecv < 0) printf("Error receiving packet\n");
        printf("%s\n", recv_buf);
        strcat(messagebox, changeline);
        strcat(messagebox, recv_buf);

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

    }
    else{

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

10. 而 client 端這裡做的事情，就是先接收 server 的訊息印出後，再等待 user 進行 I/O，接著再把獨到的訊息存進 send_buf 裡回傳給 sever。

```c
while(1){
    bytesRecv = recv(ser_socket, 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);

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

}
```

之後就可以 print 出如 spec 裡要求的訊息了！

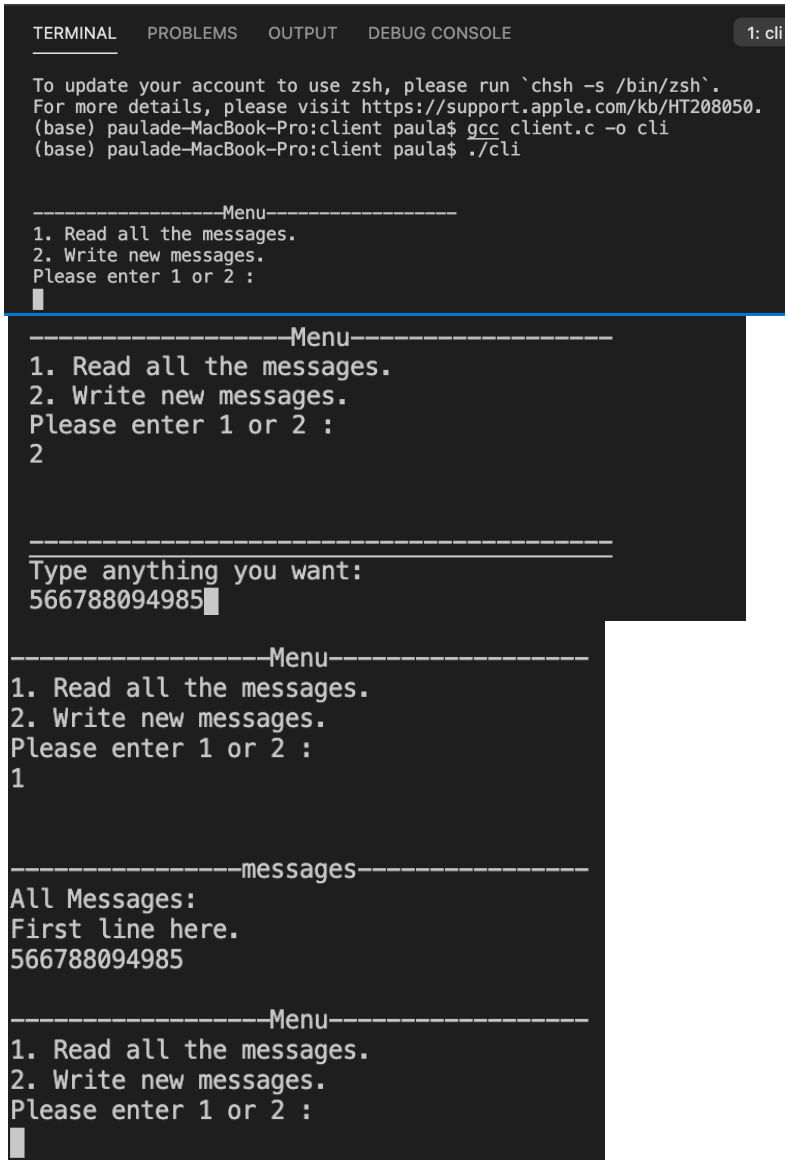
結果：

Sever 端

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) paulade-MacBook-Pro:sever paula$ gcc server.c -o ser
(base) paulade-MacBook-Pro:sever paula$ ./ser
Waiting to the client!
Client connect successfully
```

Client 端



Part.III difficulties:

其實我認為照著助教給的 socket tutorial 就能很明白這次作業的架構，再加上之前的提示 code，和網路上許多關於 socket 的簡介文章，都能幫助我很快抓住這次實作的重點，所以沒有遇到太到到無法解決的困難。

學期將至，感謝老師和助教的辛苦付出～