

兩個聊天室各有兩個使用者的執行結果如下。

三、 程式碼

環境

- CMake 3.17.5
- MinGW w64 6.0
- GCC 8.1.0
- CLion 2021.1

[chat_server.cpp]

Github:

https://github.com/linwebs/network_programming/blob/main/chat_server_v3.cpp

```

/*
 * NCYU 109 Network Programming chat server v3
 * Created by linwebs on 2021/4/12.
 */

#include <iostream>
#include <cstring>
#include <winsock.h>

#define MAXLINE 1024

using namespace std;

struct translate_word {
    string english;
    string chinese;

```

```

};

struct bad_word {
    string origin;
    string cover;
};

const int translate_word_num = 4;
const int bad_word_num = 4;

translate_word translate_word_list[translate_word_num] = {
    {"hello",      "你好"},
    {"hi",         "嗨"},
    {"how are you", "你好嗎"},
    {"I'm find",   "我很好"}
};

bad_word bad_word_list[bad_word_num] = {
    {"bullshit", "bullsxxt"},
    {"bitch",    "bixxh"},
    {"fuck",     "fxxk"},
    {"suck",     "sxxk"}
};

string cover_bad(const string &input);

string translate(const string &input);

string get_user(const string &input);

string get_room(const string &input);

string get_msg(const string &input);

int main() {
    // client count
    int cli_num = 3;

```

```
// chat room name
string now_room;
string now_user;
string now_msg;

string tmp;

cout << "歡迎使用聊天室伺服器" << endl;
cout << "請輸入使用人數: ";
cin >> cli_num;

SOCKET serv_sd;
SOCKET clis_sd;
SOCKET cli_sd[cli_num];

int clis_len;
int cli_len[cli_num];

char str_recv[MAXLINE];
char str_send[MAXLINE];

sockaddr_in serv{};
sockaddr_in clis{};
sockaddr_in cli[cli_num];

WSADATA wsadata;

// server's ip address
const char server_ip[16] = "127.0.0.1";

// server's port number
u_short server_port = 5678;

// receive bytes
int rec_len;

// send bytes
```

```

int send_len;

// bind status
int bind_status;

// Call WSASStartup() to Register "WinSock DLL"
WSASStartup(0x101, (LPWSADATA) &wsadata);

// Open a TCP socket
serv_sd = socket(AF_INET, SOCK_STREAM, 0);

// Prepare for connect.
// Include sockaddr_in struct (serv)
serv.sin_family = AF_INET;

// server's ip address
serv.sin_addr.s_addr = inet_addr(server_ip);

// server's port number
// htons: host to network
serv.sin_port = htons(server_port);

// bind
bind_status = bind(serv_sd, (LPSOCKADDR) &serv, sizeof(serv));

if (bind_status == SOCKET_ERROR) {
    cout << "bind function failed with error: " << WSAGetLastError()
<< endl;
    closesocket(serv_sd);
    WSACleanup();
    return 1;
}

// call listen() function to let socket enter listen mode
listen(serv_sd, 5);

for (int i = 0; i < cli_num; i++) {
    cli_len[i] = sizeof(cli[i]);
}

```

```

// accept connect
cout << "[訊息] 等待 client " << i + 1 << " 連線" << endl;
cli_sd[i] = accept(serv_sd, (LPSOCKADDR) &cli[i], &cli_len[i]);
cout << "[訊息] client " << i + 1 << " 已連線" << endl;
}

clis_len = sizeof(clis);
while (true) {
    clis_sd = accept(serv_sd, (LPSOCKADDR) &clis, &clis_len);

    if (clis_sd == SOCKET_ERROR) {
        cout << "[錯誤] 無法接受訊息" << endl;
        cout << "get hp error, code:" << WSAGetLastError() << endl;
        break;
    }

    //cout << "[訊息] client 已連線" << endl;

    rec_len = recv(clis_sd, str_recv, MAXLINE, 0);

    if (rec_len <= 0) {
        cout << "[錯誤] 接收訊息錯誤" << endl;
        break;
    }

    closesocket(clis_sd);

    //cout << "[接收] " << str_recv << endl;

    now_room = get_room(str_recv);
    now_user = get_user(str_recv);
    now_msg = get_msg(str_recv);

    cout << "[room]: " << now_room << "[user]: " << now_user << "[msg] : " << now_msg << endl;

    // cover bad text

```

```

tmp = cover_bad(string(now_msg));

if (!tmp.empty()) {
    now_msg = tmp;
}

// translate
tmp = translate(string(now_msg));

if (!tmp.empty()) {
    now_msg = tmp;
}

tmp = now_user;
tmp.append(",");
tmp.append(now_room);
tmp.append(",");
tmp.append(now_msg);

strcpy(str_send, tmp.c_str());

for (int i = 0; i < cli_num; i++) {
    // send msg from server to client 1
    send_len = send(cli_sd[i], str_send, int(strlen(str_send) + 1),
0);

    if (send_len == SOCKET_ERROR) {
        cout << "[錯誤] 傳送訊息到" << i << "失敗" << endl;
    }
}

if (now_msg == "bye") {
    break;
}
}

// close TCP socket
closesocket(serv_sd);

```

```

closesocket(clis_sd);
for (int i = 0; i < cli_num; i++) {
    closesocket(cli_sd[i]);
}

// finish "WinSock DLL"
WSACleanup();

//system("pause");

return 0;
}

string cover_bad(const string &input) {
    for (auto &i : bad_word_list) {
        if (input == i.origin) {
            return i.cover;
        }
    }
    return "";
}

string translate(const string &input) {
    for (auto &i : translate_word_list) {
        if (input == i.english) {
            return "(英文 -> 中文) 原文: " + input + " 譯文: " + i.chinese;
        } else if (input == i.chinese) {
            return "(中文 -> 英文) 原文: " + input + " 譯文: " + i.english;
        }
    }
    return "";
}

string get_user(const string &input) {
    return input.substr(0, input.find(','));
}

string get_room(const string &input) {

```



```

    // no room
    string no = input.substr(input.find(',') + 1);
    return no.substr(0, no.find(','));
}

```

```

string get_msg(const string &input) {
    string no;
    // no room
    no = input.substr(input.find(',') + 1);
    // no user
    no = no.substr(no.find(',') + 1);
    return no.substr(0, no.find(','));
}

```

[chat_client_send.cpp]

Github:

https://github.com/linwebs/network_programming/blob/main/chat_client_send_v3.cpp

```

/*
 * NCYU 109 Network Programming chat client send v3
 * Created by linwebs on 2021/4/12.
 */
#include <iostream>
#include <winsock.h>

#define MAXLINE 1024

using namespace std;

int main() {
    SOCKET sd;
    struct sockaddr_in serv{};

    // input msg
    string input;

    // user name

```

```

string user;

// room name
string room;

// separate symbol
string separate = ",";

char str[MAXLINE] = "";

WSADATA wsadata;

// server's ip address
const char server_ip[16] = "127.0.0.1";

// server's port number
u_short server_port = 5678;

// connect status
int conn_status;

cout << "歡迎使用聊天室 client 端傳送程式" << endl;

cout << "請輸入使用者名稱: ";
getline(cin, user);
cout << endl;

cout << "請輸入房間名稱: ";
getline(cin, room);
cout << endl;

// Call WSStartup() to Register "WinSock DLL"
WSStartup(0x101, (LPWSADATA) &wsadata);

// Prepare for connect.
// Include sockaddr_in struct (serv)
serv.sin_family = AF_INET;

```

```

// server's ip address
serv.sin_addr.s_addr = inet_addr(server_ip);

// server's port number
// htons: host to network
serv.sin_port = htons(server_port);

while (true) {
    // Open a TCP socket
    sd = socket(AF_INET, SOCK_STREAM, 0);

    cout << "[輸入] 請輸入文字: (輸入 bye 離開程式)" << endl;

    getline(cin, input);

    // connect to server
    conn_status = connect(sd, (LPSOCKADDR) &serv, sizeof(serv));

    if (conn_status == SOCKET_ERROR) {
        cout << "connect function failed with error: " <<
WSAGetLastError() << endl;
        closesocket(sd);
        WSACleanup();
        return 1;
    }
    strcpy(str, user.c_str());
    strcat(str, separate.c_str());
    strcat(str, room.c_str());
    strcat(str, separate.c_str());
    strcat(str, input.c_str());

    // send to server
    send(sd, str, int(strlen(str) + 1), 0);
    cout << "[" << user << "於" << room << "傳送] : " << input << endl;

    cout << "-----" << endl;

    if (input == "bye") {

```

```

        break;
    }
}

// close TCP socket
closesocket(sd);

// finish "WinSock DLL"
WSACleanup();

//system("pause");

return 0;
}

```

[chat_client_recv.cpp]

Github:

https://github.com/linwebs/network_programming/blob/main/chat_client_recv_v3.cpp

```

/*
 * NCYU 109 Network Programming chat client recv v3
 * Created by linwebs on 2021/4/12.
 */
#include <iostream>
#include <winsock.h>

#define MAXLINE 1024

using namespace std;

string get_user(const string &input);

string get_room(const string &input);

string get_msg(const string &input);

```

```
int main() {
    SOCKET sd;
    struct sockaddr_in serv{};

    // user name
    string user;
    string now_room;
    string now_user;
    string now_msg;

    // room name
    string room;

    char str[MAXLINE] = "";

    WSADATA wsadata;

    // server's ip address
    const char server_ip[16] = "127.0.0.1";

    // server's port number
    u_short server_port = 5678;

    // receive bytes
    int rec_len;

    // connect status
    int conn_status;

    cout << "歡迎使用聊天室 client 端顯示程式" << endl;

    cout << "請輸入使用者名稱: ";
    getline(cin, user);
    cout << endl;

    cout << "請輸入房間名稱: ";
    getline(cin, room);
    cout << endl;
```

```

// Call WSASStartup() to Register "WinSock DLL"
WSASStartup(0x101, (LPWSADATA) &wsadata);

// Open a TCP socket
sd = socket(AF_INET, SOCK_STREAM, 0);

// Prepare for connect.
// Include sockaddr_in struct (serv)
serv.sin_family = AF_INET;

// server's ip address
serv.sin_addr.s_addr = inet_addr(server_ip);

// server's port number
// htons: host to network
serv.sin_port = htons(server_port);

// connect to server
conn_status = connect(sd, (LPSOCKADDR) &serv, sizeof(serv));

if (conn_status == SOCKET_ERROR) {
    cout << "connect function failed with error: " << WSAGetLastError()
<< endl;
    closesocket(sd);
    WSACleanup();
    return 1;
}

while (true) {
    // receive from server
    rec_len = recv(sd, str, MAXLINE, 0);

    if (rec_len <= 0) {
        cout << "[錯誤] 接收訊息錯誤" << endl;
        break;
    }
}

```

```

// 接收並印出訊息

// room is correct

now_user = get_user(str);
now_room = get_room(str);
now_msg = get_msg(str);
//cout << "[user]: " << now_user << "[room]: " << now_room <<
"[msg] : " << now_msg << endl;

if (now_room == room) {
    if (now_user == user) {
        // my self
        cout << "[我自己]:\t" << get_msg(str) << endl;
    } else {
        // other people
        cout << "[" << get_user(str) << "]:\t" << get_msg(str) <<
endl;
    }
}
//cout << "[訊息]: " << str << endl;

if (now_msg == "bye") {
    break;
}
}

// close TCP socket
closesocket(sd);

// finish "WinSock DLL "
WSACleanup();

//system("pause");

return 0;
}

```

```

string get_user(const string &input) {
    return input.substr(0, input.find(','));
}

string get_room(const string &input) {
    // no room
    string no = input.substr(input.find(',') + 1);
    return no.substr(0, no.find(','));
}

string get_msg(const string &input) {
    string no;
    // no room
    no = input.substr(input.find(',') + 1);
    // no user
    no = no.substr(no.find(',') + 1);
    return no.substr(0, no.find(','));
}

```

四、心得

這次的課程老師採用預先錄製影片的方式來教學，老實說，其實我不太喜歡這樣，因為我覺得這樣就失去了我到課堂上課的意義，我學新的東西喜歡聽真人教學，這樣學習成效會比較好，不過老師考量到大家的學習進度不一，這也是一種方式，只不過，我覺得要以這樣的方式上課，不如就直接讓我們遠距教學，因為看影片時，我總是會分心去邊做其他的事，反而無法專注在學習上。

這次的作業，是寫聊天室的部分，我覺得非常的實用，畢竟聊天室就是基礎，可做為將來撰寫任何程式時，做為修改的範本，當然老師提了非常多的延伸功能讓我們進行練習，不過我依舊沒辦法在課堂上完成，還是得花費課後時間將程式逐漸撰寫出來，老師的要求是三人交談，不過我覺得這樣子其實沒什麼太大的意義，於是我就研究了很久，將 server 端所有發送訊息的程式碼，改寫為動態生成，達到輸入 n 個，就可以建立 n 個使用者連線的聊天室軟體，真的費了非常多心思。