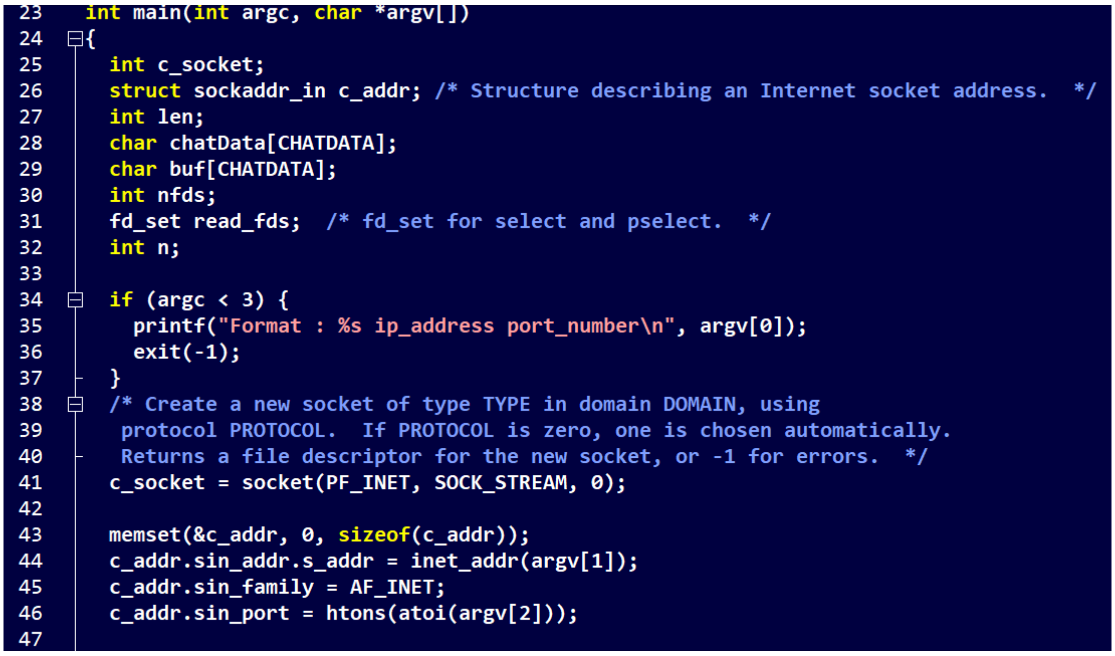
網路概論作業二　　　學號/系級/姓名　405410090/資工三/陳知余

client.c

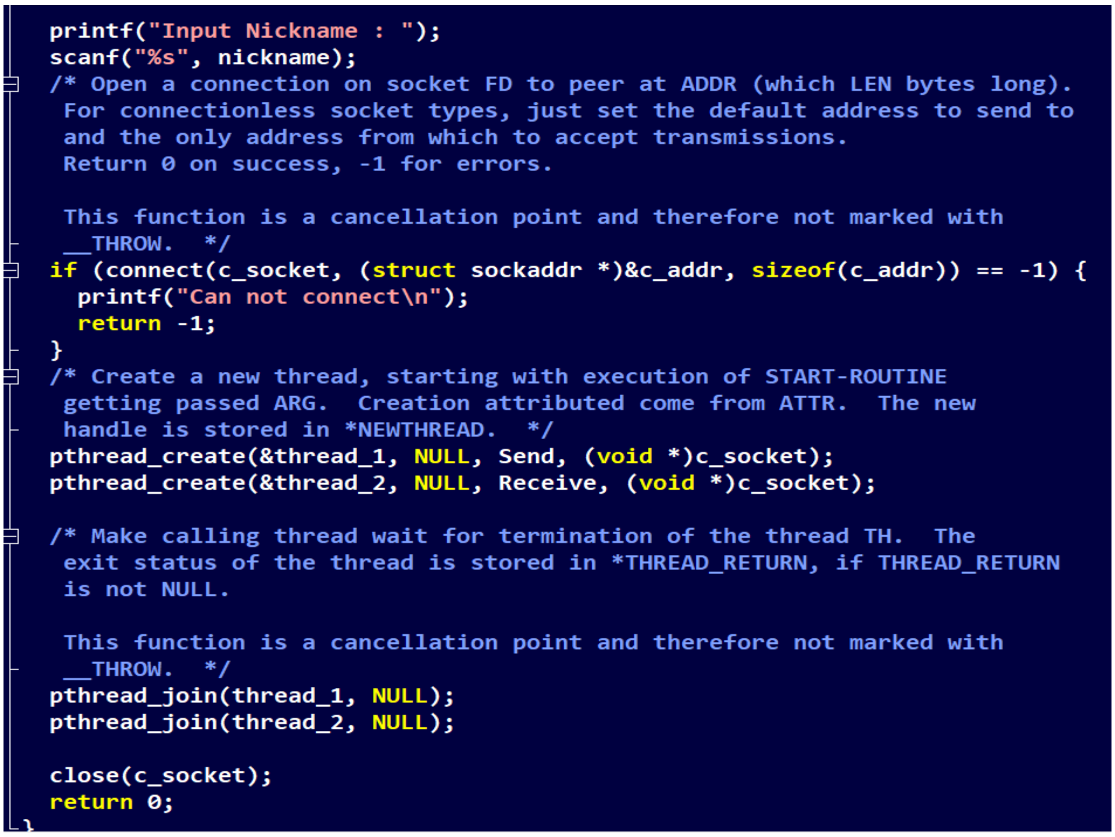
1.　程式流程圖　+ 2.功能實作技巧描述

(1)初始化，將參數讀入(ex: ./client ip port)



```c
int main(int argc, char *argv[])
{
    int c_socket;
    struct sockaddr_in c_addr; /* Structure describing an Internet socket address.  */
    int len;
    char chatData[CHATDATA];
    char buf[CHATDATA];
    int nfds;
    fd_set read_fds;  /* fd_set for select and pselect.  */
    int n;

    if (argc < 3) {
      printf("Format : %s ip_address port_number\n", argv[0]);
      exit(-1);
    }
    /* Create a new socket of type TYPE in domain DOMAIN, using
     protocol PROTOCOL.  If PROTOCOL is zero, one is chosen automatically.
     Returns a file descriptor for the new socket, or -1 for errors.  */
    c_socket = socket(PF_INET, SOCK_STREAM, 0);

    memset(&c_addr, 0, sizeof(c_addr));
    c_addr.sin_addr.s_addr = inet_addr(argv[1]);
    c_addr.sin_family = AF_INET;
    c_addr.sin_port = htons(atoi(argv[2]));

```

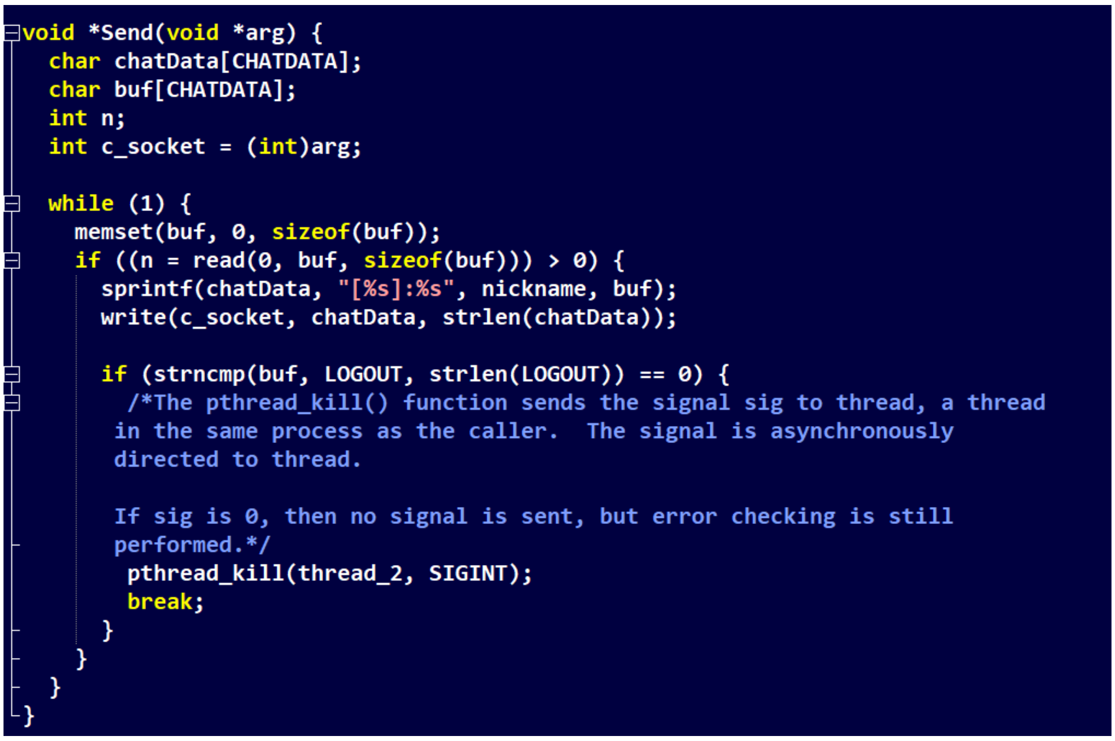
(2)首先在連線前先讓使用者輸入想使用的暱稱，可以在連上後傳給 server。接著 create 兩個 pthread，一個傳送訊息，另一個用來接收訊息。



```c
  printf("Input Nickname : ");
  scanf("%s", nickname);
  /* Open a connection on socket FD to peer at ADDR (which LEN bytes long).
   For connectionless socket types, just set the default address to send to
   and the only address from which to accept transmissions.
   Return 0 on success, -1 for errors.

   This function is a cancellation point and therefore not marked with
   __THROW.  */
  if (connect(c_socket, (struct sockaddr *)&c_addr, sizeof(c_addr)) == -1) {
    printf("Can not connect\n");
    return -1;
  }
  /* Create a new thread, starting with execution of START-ROUTINE
   getting passed ARG.  Creation attributed come from ATTR.  The new
   handle is stored in *NEWTHREAD.  */
  pthread_create(&thread_1, NULL, Send, (void *)c_socket);
  pthread_create(&thread_2, NULL, Receive, (void *)c_socket);

  /* Make calling thread wait for termination of the thread TH.  The
   exit status of the thread is stored in *THREAD_RETURN, if THREAD_RETURN
   is not NULL.

   This function is a cancellation point and therefore not marked with
   __THROW.  */
  pthread_join(thread_1, NULL);
  pthread_join(thread_2, NULL);

  close(c_socket);
  return 0;
}
```

(3)傳送訊息的函式(Send)：

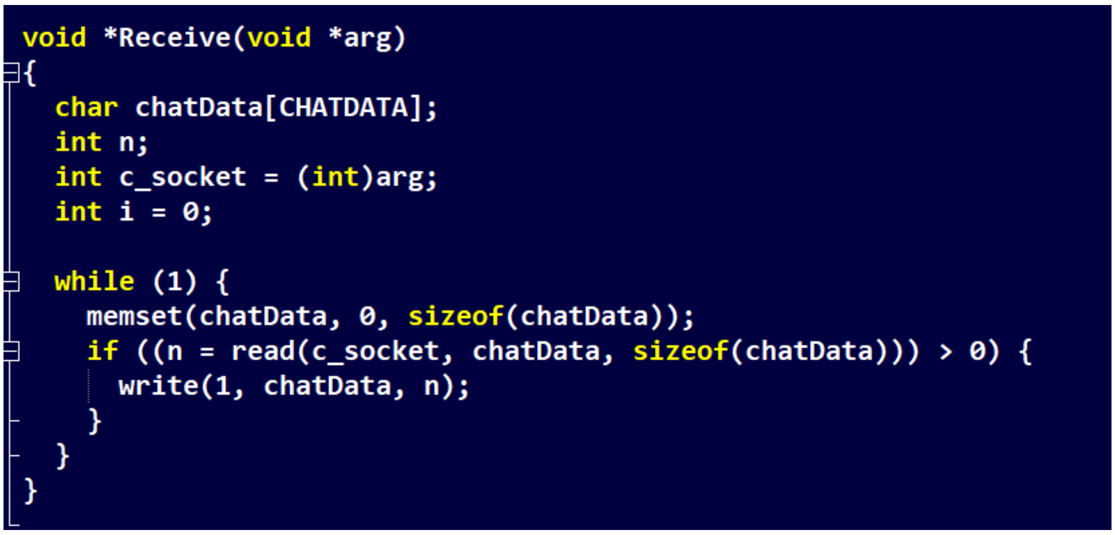
先將 buf 用 memset 清零，接著如果有讀入東西，都預設在前頭以左右中括號加上自己的暱稱，寫入 socket 裡頭，假設發現輸入的指令要登出時(LOGOUT[] = “!exit”)，將 receive 的 thread 殺掉，自己也 break 跳出 while loop。



```
void *Send(void *arg) {
  char chatData[CHATDATA];
  char buf[CHATDATA];
  int n;
  int c_socket = (int)arg;

  while (1) {
    memset(buf, 0, sizeof(buf));
    if ((n = read(0, buf, sizeof(buf))) > 0) {
      sprintf(chatData, "[%s]:%s", nickname, buf);
      write(c_socket, chatData, strlen(chatData));

      if (strncmp(buf, LOGOUT, strlen(LOGOUT)) == 0) {
        /*The pthread_kill() function sends the signal sig to thread, a thread
       in the same process as the caller.  The signal is asynchronously
       directed to thread.

       If sig is 0, then no signal is sent, but error checking is still
       performed.*/
        pthread_kill(thread_2, SIGINT);
        break;
      }
    }
  }
}
```

(4)接收訊息的函式(Receive)：

將 chatData 清零，假如從 socket 有讀到東西，就將它寫入 chatData 裡頭。



```
void *Receive(void *arg)
{
  char chatData[CHATDATA];
  int n;
  int c_socket = (int)arg;
  int i = 0;

  while (1) {
    memset(chatData, 0, sizeof(chatData));
    if ((n = read(c_socket, chatData, sizeof(chatData))) > 0) {
      write(1, chatData, n);
    }
  }
}
```

3. 重要資料結構：

struct sockaddr_in {

sa_family_t   sin_family; /* address family: AF_INET */

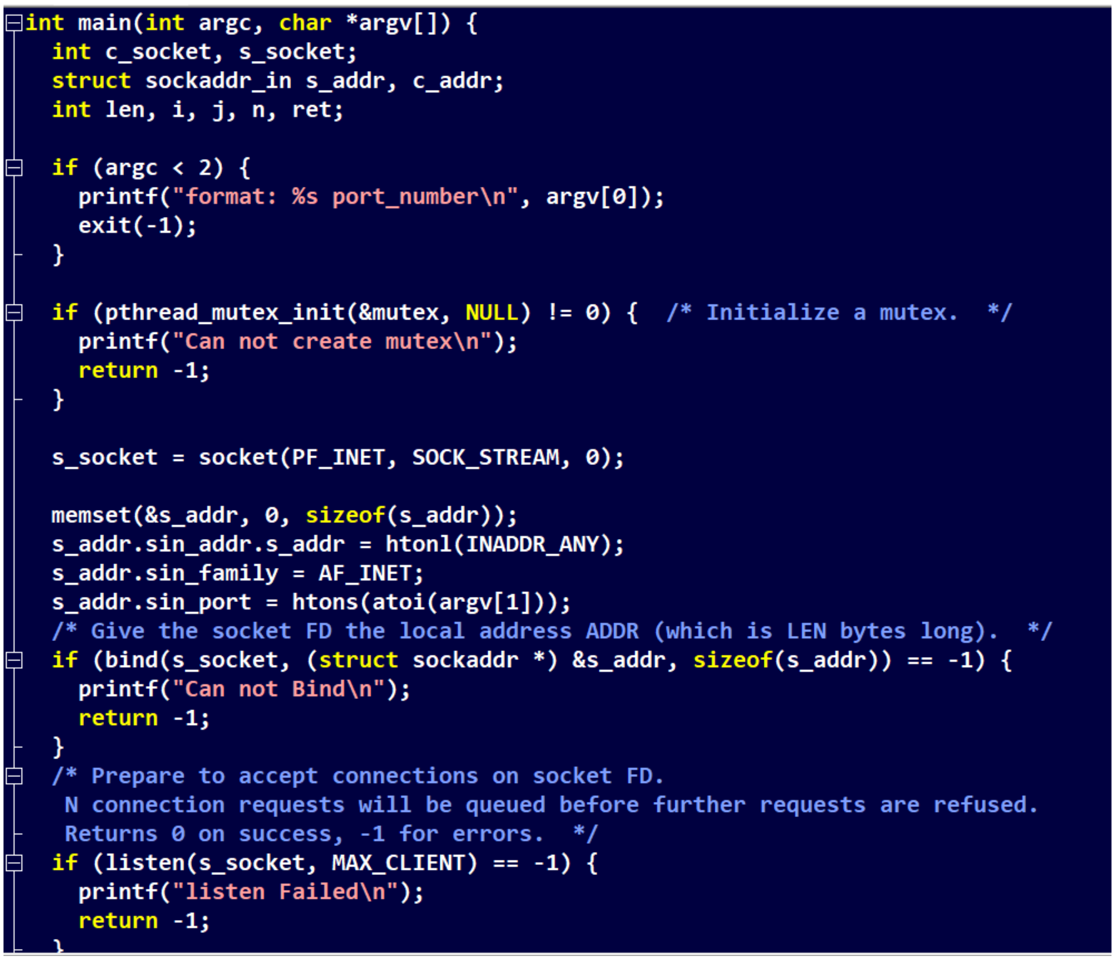
in_port_t   sin_port;   /* port in network byte order */

```
        struct in_addr sin_addr;      /* internet address */
};

/* Internet address. */
struct in_addr {
        uint32_t           s_addr;        /* address in network byte order */
};
```

Server.c

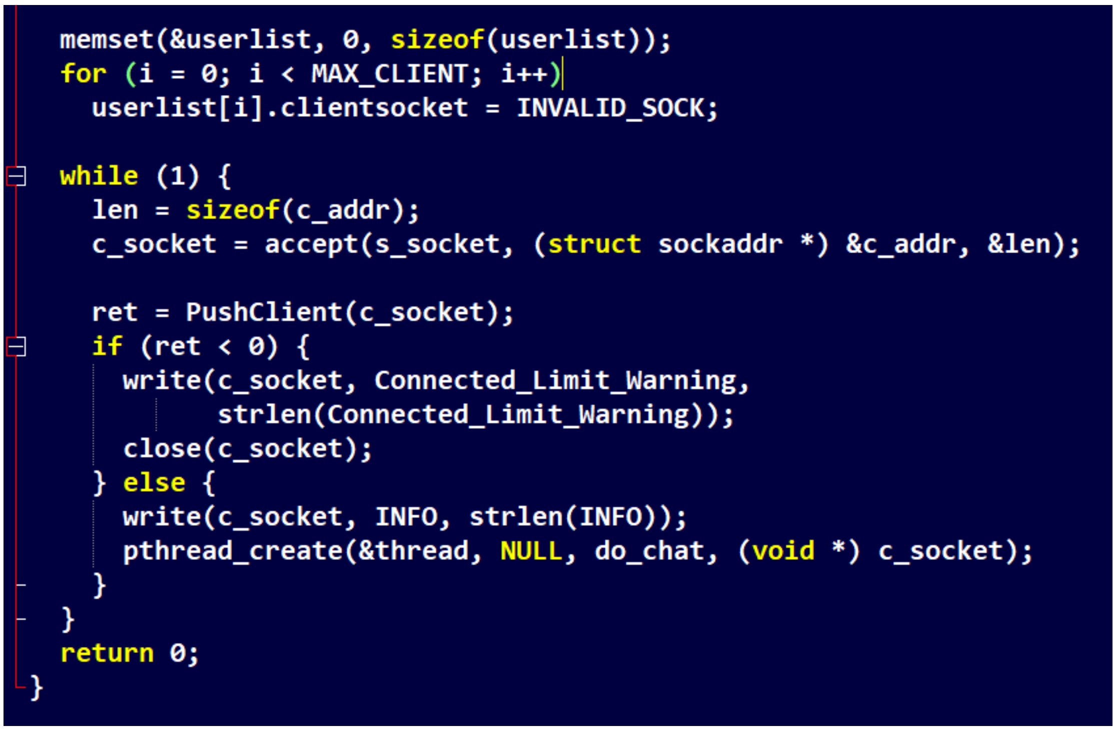
1. 程式流程圖

(1) 首先檢查參數，預設要是./server port_number，接著設定好 s_socket、sockaddr 之後，依序使用 bind 和 listen 函式。



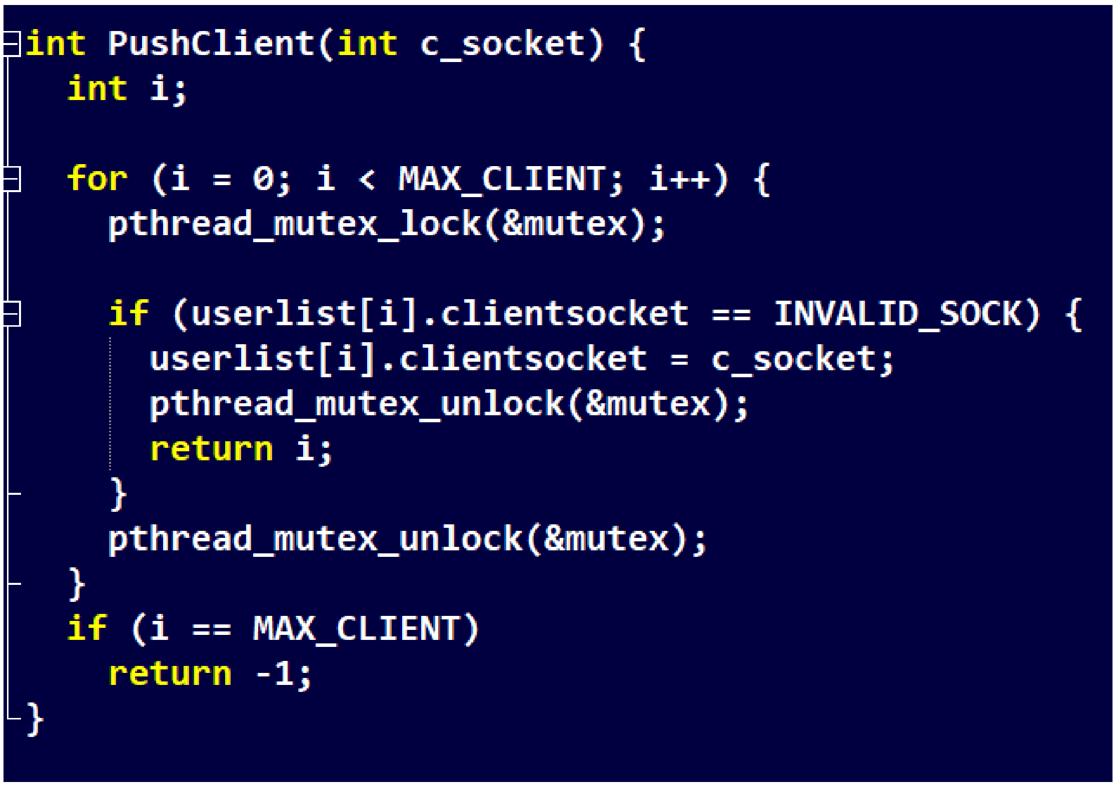
```
int main(int argc, char *argv[]) {
  int c_socket, s_socket;
  struct sockaddr_in s_addr, c_addr;
  int len, i, j, n, ret;

  if (argc < 2) {
    printf("format: %s port_number\n", argv[0]);
    exit(-1);
  }

  if (pthread_mutex_init(&mutex, NULL) != 0) {  /* Initialize a mutex.  */
    printf("Can not create mutex\n");
    return -1;
  }

  s_socket = socket(PF_INET, SOCK_STREAM, 0);

  memset(&s_addr, 0, sizeof(s_addr));
  s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  s_addr.sin_family = AF_INET;
  s_addr.sin_port = htons(atoi(argv[1]));
  /* Give the socket FD the local address ADDR (which is LEN bytes long).  */
  if (bind(s_socket, (struct sockaddr *) &s_addr, sizeof(s_addr)) == -1) {
    printf("Can not Bind\n");
    return -1;
  }
  /* Prepare to accept connections on socket FD.
   N connection requests will be queued before further requests are refused.
   Returns 0 on success, -1 for errors.  */
  if (listen(s_socket, MAX_CLIENT) == -1) {
    printf("listen Failed\n");
    return -1;
```

(2) 接著將使用者名單清零，進入 while loop 開始用 accept 接受 client 的 connect，接著使用 PushClient 將連線的 client 存入尚未使用的 userlist，假如回傳負值代表達到上限要警告 client，回傳正值則把功能介紹的 INFO 字串寫入給 client socket，create 一個 thread。



```
  memset(&userlist, 0, sizeof(userlist));
  for (i = 0; i < MAX_CLIENT; i++)
    userlist[i].clientsocket = INVALID_SOCK;

  while (1) {
    len = sizeof(c_addr);
    c_socket = accept(s_socket, (struct sockaddr *) &c_addr, &len);

    ret = PushClient(c_socket);
    if (ret < 0) {
      write(c_socket, Connected_Limit_Warning,
            strlen(Connected_Limit_Warning));
      close(c_socket);
    } else {
      write(c_socket, INFO, strlen(INFO));
      pthread_create(&thread, NULL, do_chat, (void *) c_socket);
    }
  }
  return 0;
}
```

(PushClient)

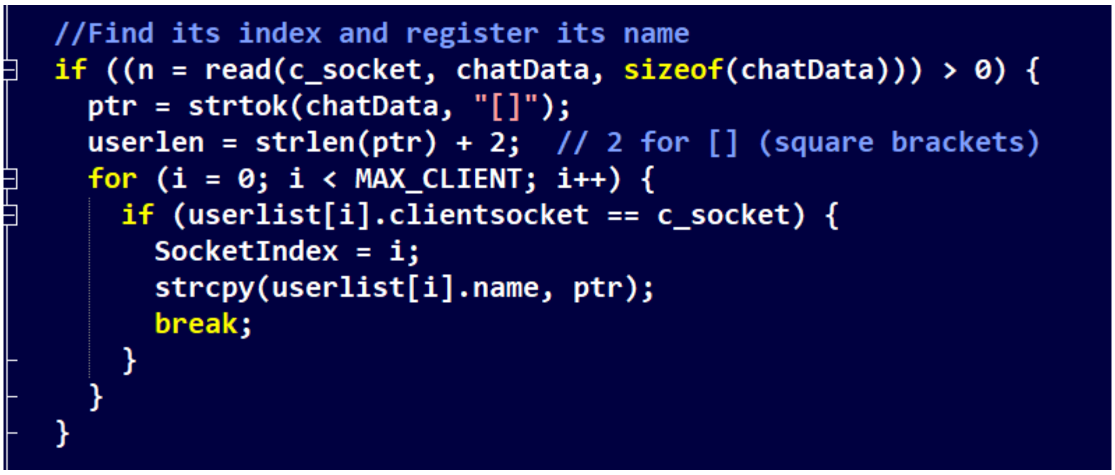


```
int PushClient(int c_socket) {
  int i;

  for (i = 0; i < MAX_CLIENT; i++) {
    pthread_mutex_lock(&mutex);

    if (userlist[i].clientsocket == INVALID_SOCK) {
      userlist[i].clientsocket = c_socket;
      pthread_mutex_unlock(&mutex);
      return i;
    }
    pthread_mutex_unlock(&mutex);
  }
  if (i == MAX_CLIENT)
    return -1;
}
```

2. 函式功能描述

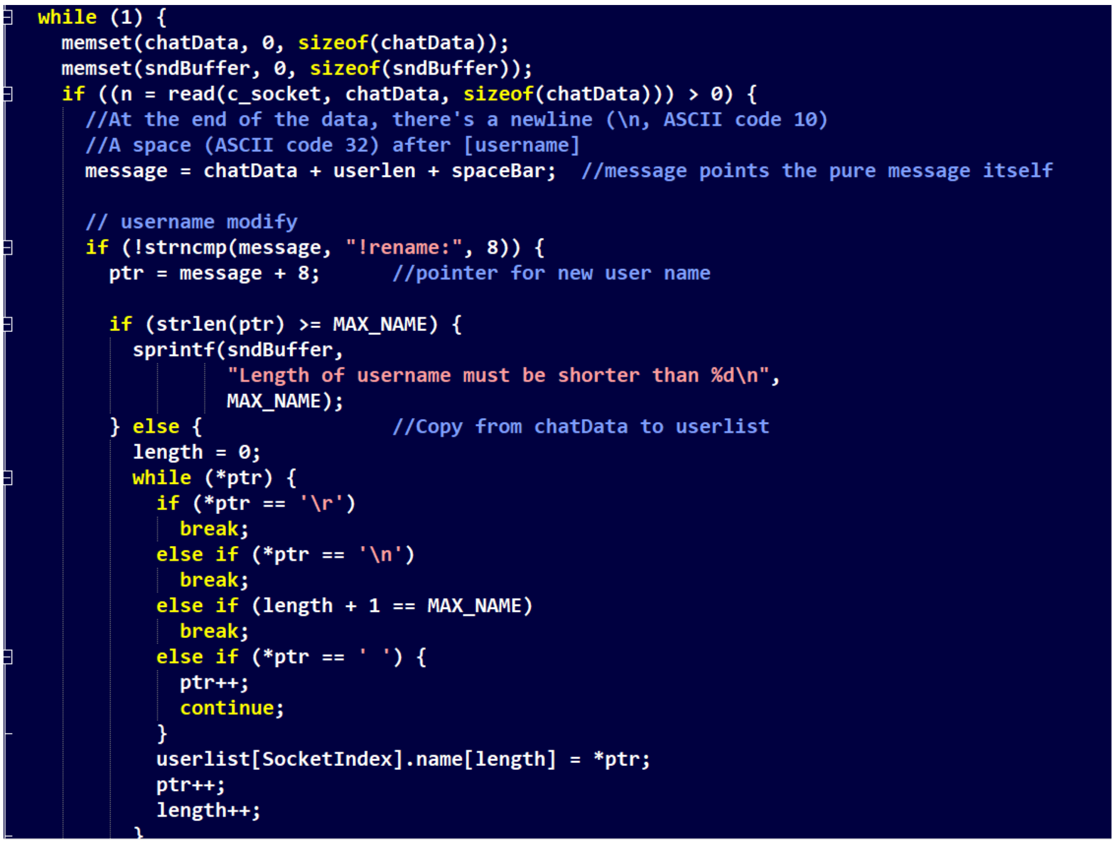
(1) do_chat：主要的聊天功能函式

當從 client socket 中第一次讀到訊息時，就是 client 一開始輸入的暱稱，從 user list 根據存取的 socketfd 找到對應的使用者，strcpy 過去。



```
//Find its index and register its name
if ((n = read(c_socket, chatData, sizeof(chatData))) > 0) {
  ptr = strtok(chatData, "[]");
  userlen = strlen(ptr) + 2;  // 2 for [] (square brackets)
  for (i = 0; i < MAX_CLIENT; i++) {
    if (userlist[i].clientsocket == c_socket) {
      SocketIndex = i;
      strcpy(userlist[i].name, ptr);
      break;
    }
  }
}
```

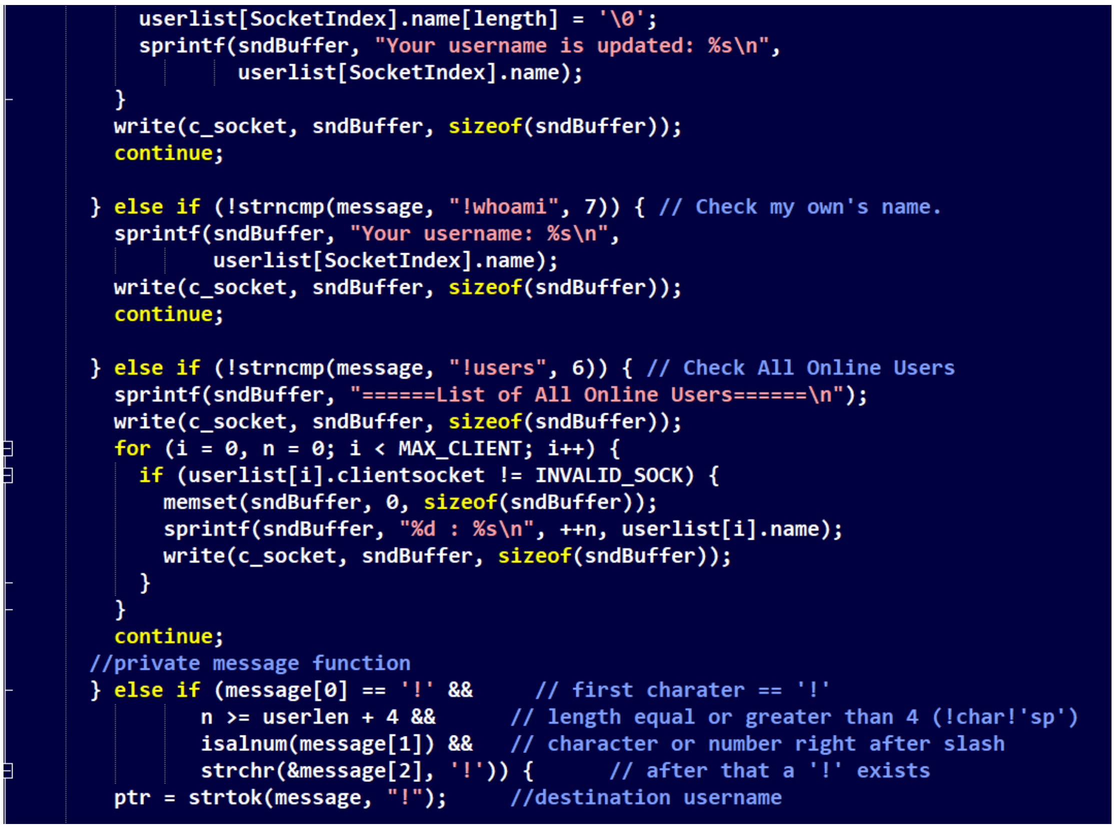
接著進入 while 迴圈，先將 chatData 和 sndBuffer 清零，首先第一個 if 是改名指令!rename 的 implement，首先檢查是否超過 user name 的最大值，接著排除其他空白字元，找到實際輸入的名稱，賦予該 socketfd 的姓名。



```
while (1) {
  memset(chatData, 0, sizeof(chatData));
  memset(sndBuffer, 0, sizeof(sndBuffer));
  if ((n = read(c_socket, chatData, sizeof(chatData))) > 0) {
    //At the end of the data, there's a newline (\n, ASCII code 10)
    //A space (ASCII code 32) after [username]
    message = chatData + userlen + spaceBar;  //message points the pure message itself

    // username modify
    if (!strncmp(message, "!rename:", 8)) {
      ptr = message + 8;      //pointer for new user name

      if (strlen(ptr) >= MAX_NAME) {
        sprintf(sndBuffer,
                "Length of username must be shorter than %d\n",
                MAX_NAME);
      } else {                //Copy from chatData to userlist
        length = 0;
        while (*ptr) {
          if (*ptr == '\r')
            break;
          else if (*ptr == '\n')
            break;
          else if (length + 1 == MAX_NAME)
            break;
          else if (*ptr == ' ') {
            ptr++;
            continue;
          }
          userlist[SocketIndex].name[length] = *ptr;
          ptr++;
          length++;
        }
```

接著下個 else if 裡是實作!whoami 指令，可以顯示自己的暱稱，由 userlist 的 client socketfd 的索引就能輸出姓名。

下個 else if 裡是實作!users 指令，用以顯示所有在線使用者，開一個 for loop 從 userlist 找，n 表示編號，只要不等於 INVALID_SOCK(-1)的就代表有人，可以輸出。



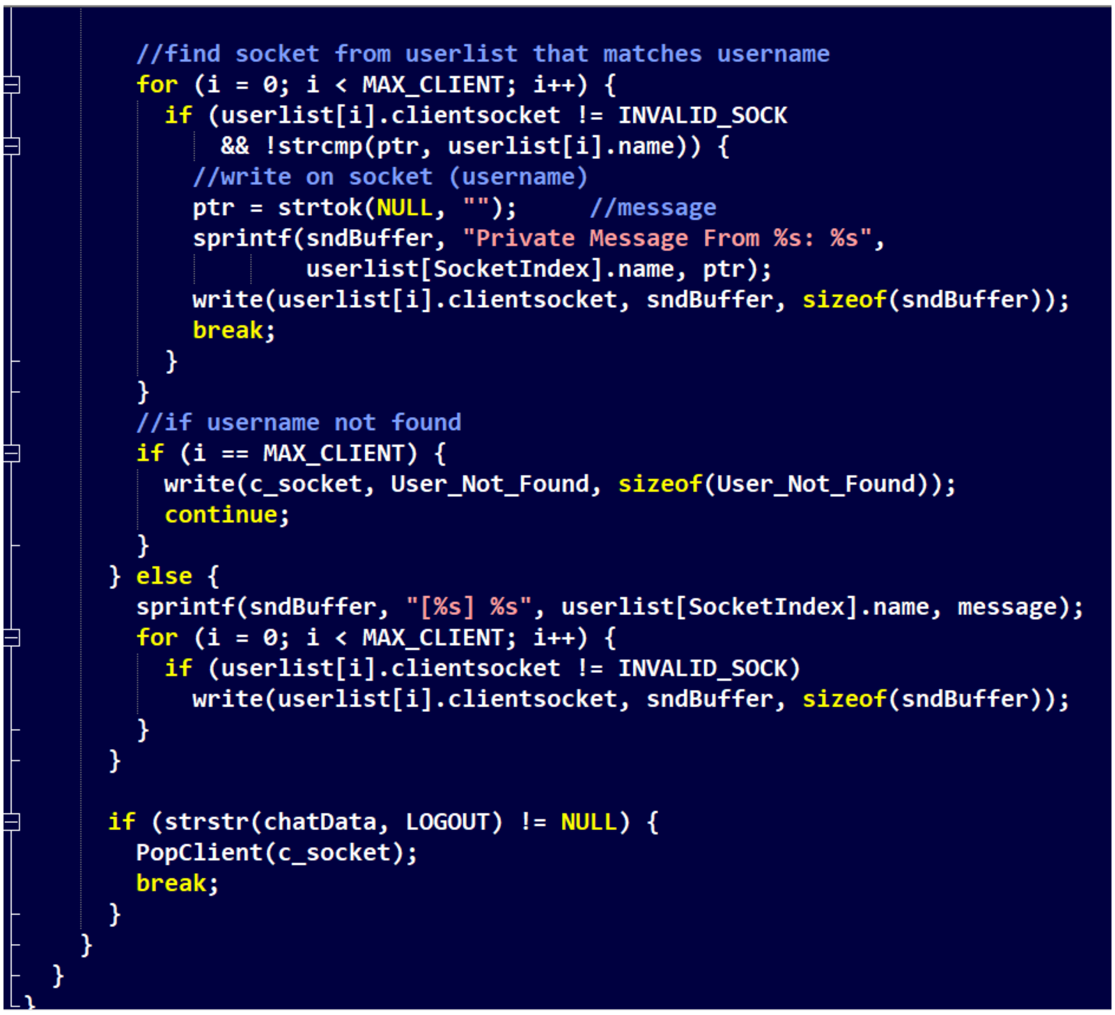
```
        userlist[SocketIndex].name[length] = '\0';
        sprintf(sndBuffer, "Your username is updated: %s\n",
                userlist[SocketIndex].name);
      }
      write(c_socket, sndBuffer, sizeof(sndBuffer));
      continue;

    } else if (!strncmp(message, "!whoami", 7)) { // Check my own's name.
      sprintf(sndBuffer, "Your username: %s\n",
              userlist[SocketIndex].name);
      write(c_socket, sndBuffer, sizeof(sndBuffer));
      continue;

    } else if (!strncmp(message, "!users", 6)) { // Check All Online Users
      sprintf(sndBuffer, "======List of All Online Users======\n");
      write(c_socket, sndBuffer, sizeof(sndBuffer));
      for (i = 0, n = 0; i < MAX_CLIENT; i++) {
        if (userlist[i].clientsocket != INVALID_SOCK) {
          memset(sndBuffer, 0, sizeof(sndBuffer));
          sprintf(sndBuffer, "%d : %s\n", ++n, userlist[i].name);
          write(c_socket, sndBuffer, sizeof(sndBuffer));
        }
      }
      continue;
    //private message function
    } else if (message[0] == '!' &&     // first charater == '!'
               n >= userlen + 4 &&      // length equal or greater than 4 (!char!'sp')
               isalnum(message[1]) &&   // character or number right after slash
               strchr(&message[2], '!')) {      // after that a '!' exists
      ptr = strtok(message, "!");     //destination username
```

下一個則是實作私訊的方程式，預設格式是要用兩個!隔開 (ex: !(username)!)，使用 strtok 擷取出來後，使用 for loop 進入 userlist 尋找同時有 socketfd 和姓名符合的使用者，將訊息寫給對方。

假如 i == MAX_CLIENT，代表跑到迴圈結尾沒有符合的人名，便傳送警語通知查無此人。

最後如果都沒有以!提示的指令，那就是預設的 public message，以 for loop 檢查，只要 fd 不是 INVALID，就傳給他。

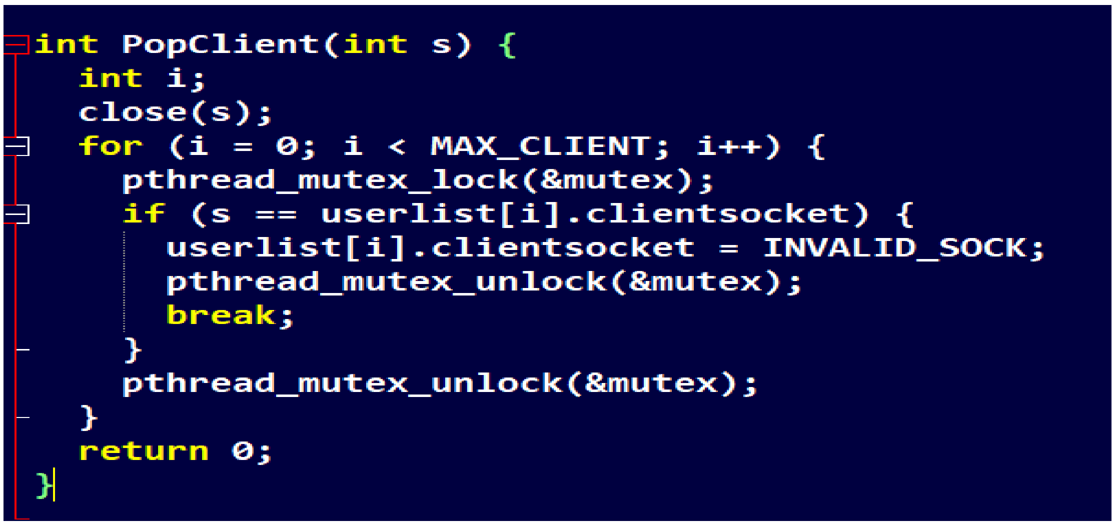
假如使用者傳送!exit，呼叫 PopClient 函式。



```
    //find socket from userlist that matches username
    for (i = 0; i < MAX_CLIENT; i++) {
     if (userlist[i].clientsocket != INVALID_SOCK
       && !strcmp(ptr, userlist[i].name)) {
      //write on socket (username)
      ptr = strtok(NULL, "");     //message
      sprintf(sndBuffer, "Private Message From %s: %s",
          userlist[SocketIndex].name, ptr);
      write(userlist[i].clientsocket, sndBuffer, sizeof(sndBuffer));
      break;
     }
    }
    //if username not found
    if (i == MAX_CLIENT) {
     write(c_socket, User_Not_Found, sizeof(User_Not_Found));
     continue;
    }
   } else {
    sprintf(sndBuffer, "[%s] %s", userlist[SocketIndex].name, message);
    for (i = 0; i < MAX_CLIENT; i++) {
     if (userlist[i].clientsocket != INVALID_SOCK)
      write(userlist[i].clientsocket, sndBuffer, sizeof(sndBuffer));
    }
   }

   if (strstr(chatData, LOGOUT) != NULL) {
    PopClient(c_socket);
    break;
   }
  }
 }
}
```

(2) PopClient

參數 s 代表欲關閉的 fd，首先關掉，先將 mutex 鎖起來，找到相符的 client socket，改成 INVALID_SOCK，解開 mutex 後退出 loop。



```
int PopClient(int s) {
 int i;
 close(s);
 for (i = 0; i < MAX_CLIENT; i++) {
  pthread_mutex_lock(&mutex);
  if (s == userlist[i].clientsocket) {
   userlist[i].clientsocket = INVALID_SOCK;
   pthread_mutex_unlock(&mutex);
   break;
  }
  pthread_mutex_unlock(&mutex);
 }
 return 0;
}
```

3. 重要的資料結構

(1) struct user {

int clientsocket;　//client 的 socket

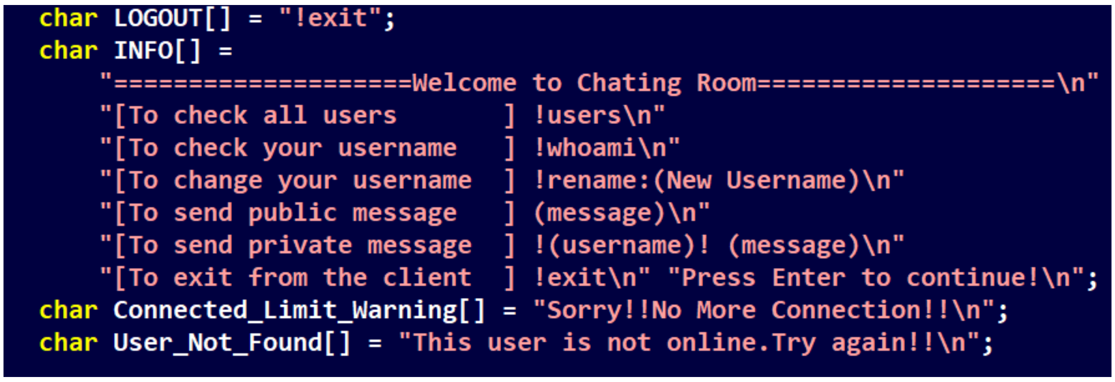
    char name[MAX_NAME]; // client 的 user name
};

(2) sockaddr_in:同 client.c

4. 期望加分的新功能或特點：

(1) 註解完整

(2) 有給 client 的指令導覽，有作業指定的查在線使用者、私訊、公共訊息，還多做了改名和登出的功能(但少做傳檔功能)



```
char LOGOUT[] = "!exit";
char INFO[] =
    "====================Welcome to Chating Room====================\n"
    "[To check all users        ] !users\n"
    "[To check your username    ] !whoami\n"
    "[To change your username   ] !rename:(New Username)\n"
    "[To send public message    ] (message)\n"
    "[To send private message   ] !(username)! (message)\n"
    "[To exit from the client   ] !exit\n" "Press Enter to continue!\n";
char Connected_Limit_Warning[] = "Sorry!!No More Connection!!\n";
char User_Not_Found[] = "This user is not online.Try again!!\n";
```