**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

LẬP TRÌNH SOCKET

*Nguyễn Hoàng Minh – 18127265  
Lê Trần Trúc Vân - 18127250*

**Môn học:** Mạng Máy Tính **Giảng viên:** ThS. Đỗ Hùng Cường  
 ThS. Chung Thùy Linh

TP. HỒ CHÍ MINH, 2020

**Mục lục**

[1. Kịch bản giao tiếp 3](#_Toc44004996)

[2. Môi trường lập trình 6](#_Toc44004997)

[3. Hướng dẫn sử dụng các tính năng 6](#_Toc44004998)

[4. Bảng phân công 9](#_Toc44004999)

[5. Đánh giá 10](#_Toc44005000)

[6. Nguồn tham khảo 11](#_Toc44005001)

1. Kịch bản **giao** tiếp

**Server**

Socket()  
Bind()  
Listen()  
Accept()  
Close()

**Client**

Socket()  
  
Connect()  
Sending()  
Receiving()

New Thread

Request/ Response

**ClientList**

Run()

* **Server**

+ Tạo socket

+ Bind socket

+ Listen: lắng nghe kết nối từ các client.

+ Accept

Sau khi accept thành công, Server tạo 1 thread mới để giao tiếp với client.

Trong hàm run():

+ getUsername() : Lấy tên của client để kiểm tra trong UserList.txt và báo về kết quả cho client là thành công hay thất bại.

+ sendUserList(): Server cập nhập danh sách client và gửi danh sách các client đang kết nối.

+ sendFileList() : Server cập nhập danh sách file và gửi danh sách các client đang kết nối.

+ Server tiếp nhận tin nhắn client lưu vào buffer.

+ FormatString(buffer): Xử lý tin nhắn Client trả về các mode

STATUS

mode 0 : CHAT TO EVERYONE

mode 1 : CHAT TO SPECIFIC CLIENT

mode 2 : LOGGED OUT

mode 3 : SEND FILE TO EVERYONE

mode 4 : SEND FILE TO SPECIFIC CLIENT

mode 5+X: DOWNLOAD FILE WITH INDEX X

+ Ngắt kết nối từ client: Client 1 muốn ngắt kết nối với Server gửi tin nhắn #, hoặc khi client đột ngột đóng chương trình, Server xử lý tin nhắn sẽ thông báo với các client khác và đóng kết nối với Client 1.

+ Ngắt kết nối từ Server: Server thoát sẽ ngắt kết nối với tất cả client.

* **Client**

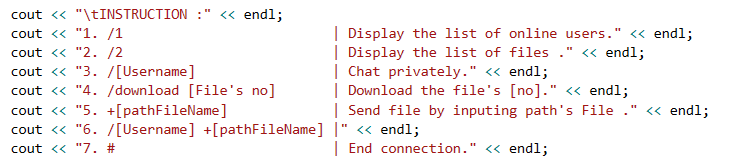
+ Socket()

+ Connect()

+ LogIn(): client gửi username và password lên server để kiểm tra thông tin và nhận lại kết quả.

Sau khi LogIn thành công, client tạo 2 thread sending và receiving

+ sending(): client nhập vào tin nhắn từ bàn phím vào sendMsg



+ getMsg(sendMsg): Tin nhắn client được phân loại và xử lý theo các cú pháp trên, sau đó truyền lên cho Server để yêu cầu từng dịch vụ phù hợp.

+ receiving():

Đầu tiên khi mới đăng nhập/ đăng ký, client nhận danh sách users và danh sách files từ Server.

Client nhận tin nhắn từ Server rồi lưu vào buffer.

+ recvMsg(buffer): Tin nhắn được nhận từ Server được phân loại và xử lý và trả kết quả lên màn hình client.

+ Ngắt kết nối: close socket và dừng chương trình.

2. Môi trường lập trình

Sử dụng thư viện Winsock2 trong Visual Studio, và các thư viện sau:

#pragma once

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <string>

#include <cstring>

#include <vector>

#include <thread>

#include <fstream>

#include <conio.h>

#include <winuser.h>

// Need to link with Ws2\_32.lib

#pragma comment (lib, "Ws2\_32.lib")

3. Hướng dẫn sử dụng các tính năng

**+ Sign Up:** Nếu chưa có tài khoản thì nhập phím 1 sau khi khởi động chương trình → nhập ID và Password → Nếu ID đã được sử dụng thì chương trình sẽ yêu cầu nhập lại ID khác. Khi Sign Up thành công chương trình sẽ tự động đăng nhập vào với tên là ID vừa mới đăng ký.

**+ Sign In:** Nếu đã có tài khoản thì nhập phím 2 sau khi khởi động chương trình → nhập ID và Password → Khi nhập sai Username hoặc Password, chương trình sẽ cho nhập lại. Nếu sai quá 3 lần thì chương trình đóng.

**+ Gửi tin nhắn chung:** client nhập tin nhắn từ bàn phím và bấm Enter.

*[tin nhắn]*

- Server: nhận được tin nhắn→ xử lý rồi gửi thông báo “+clientA: …” cho các client còn lại.

- Client: nhận tin nhắn từ server và in ra màn hình.

**+ Gửi tin nhắn riêng tư:** client gửi tin nhắn theo cú pháp

*/[username][khoảng trắng][tin nhắn]*

- Server: nhận được tin nhắn có tín hiệu chat riêng tư→ xử lý rồi gửi thông báo “+clientA to client (privately): … ” cho clientB.

- ClientB: nhận tin nhắn từ server và in ra màn hình.

**+ Gửi file chung:** client gửi tin nhắn theo cú pháp

*+[đường dẫn file]*

- Client: mở File lên để đọc kích thước và nội dung dưới binary, rồi gửi tới server.

- Server: nhận được tin nhắn có tín hiệu gửi file → xử lý rồi gửi thông báo “+clientA sent file […]” cho các client khác. Sau đó server nhận kích thước và buffer chứa nội dung file, và thêm vào fileList

- Các client khác nhận được thông báo từ server thì cập nhật là fileList của mình.

**+ Gửi file riêng tư:** client gửi tin nhắn theo cú pháp

*/[username][khoảng trắng] +[đường dẫn file]*

- Client: mở File lên để đọc kích thước và nội dung dưới binary, rồi gửi tới server.

- Server: nhận được tin nhắn có tín hiệu gửi file → xử lý rồi gửi thông báo “+clientA sent file to clientB (privately): […]” cho clientB. Sau đó server nhận kích thước và buffer chứa nội dung file, và thêm vào fileList

- Client B nhận được thông báo từ server thì cập nhật là fileList của mình.

**+ Download file:** client gửi tin nhắn theo cú pháp

*/download[khoảng trắng][thứ tự file trong FileList]*

- Server nắm danh sách tất cả các file. Server xử lý tin nhắn và thực hiện ánh xạ số thứ tự trong FileList của client vào số thứ tự trong FileList trong Server. Server gửi tên, kích thước, nội dung dưới dạng binary của file tương ứng cho client.



- Client: Client nhận tên , kích thước, nội dung dưới dạng binary của file tương ứng và mở file rồi ghi xuống.

**+ Hiện bảng hướng dẫn sử dụng:** soạn */0.*

- In câu lệnh cú pháp lên màn hình

**+ Xem danh sách các client:** soạn */1*.

- Server nắm danh sách users và gửi cho các client sau khi đăng nhập. Client lưu dữ liệu vào vector <string> onlineUser. Client sẽ in danh sách vector lên màn hình.

**+ Xem danh sách các file:** soạn */2.*

*-* Server nắm danh sách tên file và gửi cho các client sau khi đăng nhập. Client lưu dữ liệu vào vector <File> fileNameList. Client sẽ in danh sách vector lên màn hình.

**+ Sign Out:** soạn *#.*

- ClientA: gửi dấu # tới server rồi close socket và kết thúc chương trình

- Server: khi nhận được dấu # thì đóng kết nối với clientA → Server xử lý chuỗi và gửi sendMsg = “clientA logged out” tới các client còn lại. Các client còn lại nhận được tin nhắn sẽ cập nhận lại UserList.

4. Bảng phân công

* **Mức độ đóng góp từng thành viên**

|  |  |  |
| --- | --- | --- |
| **Nguyễn Hoàng Minh**  **18127265** | - Thiết kế và xử lý đa luồng bên server.  - Sign In()  - checkUser()  - FormatString()  - Hàm print trên Server  - send + receive + update UserList.  - Kiểm tra thông tin đăng nhập + đăng ký  - Thiết kế và xử lý tin nhắn ở Server  - sendFile  - Phân công và giao deadline | 50% |
| **Lê Trần Trúc Vân**  **18127250** | - Tạo môi trường giao tiếp giữa server và client.  - Thiết kế và xử lý đa luồng bên client.  - Sign Up()  - Chia hàm tạo class cho server và client  - Thiết kế và xử lý tin nhắn ở Client  - send + receive + update FileList.  - Hàm print trên Client  - getMsg() + sendMsg()  - Download file  - Viết báo cáo | 50% |

5. Đánh giá

|  |  |  |
| --- | --- | --- |
| **Chức năng** | **Mức độ hoàn thành** | **Nhận xét** |
| 1. Server phải cho phép nhiều client đăng nhập và thảo luận cùng lúc. Trên lý thuyết, số lượng client là không giới hạn. | 90% | Số lượng client là cấp phát tĩnh nên chưa đạt tới mức không giới hạn. |
| 2. Client cần đăng ký một username và password để đăng nhập. Nếu username đã được sử dụng, server sẽ yêu cầu client chọn username khác cho đến khi không bị trùng. Client sẽ sử dụng tên này trong toàn bộ quá trình chat. Tại giao diện server và client phải có mục thể hiện danh sách tất cả client đang kết nối đến. | 100% | Đạt đủ yêu cầu |
| 3. Mọi thông điệp trao đổi của client phải được thể hiện trên màn hình của server và mọi client khác | 100% | Đạt đủ yêu cầu |
| 4. Khi một client đăng nhập hoặc đăng xuất (hoặc thoát chương trình), phải có dòng thông báo trên màn hình server và mọi client. | 100% | Đạt đủ yêu cầu |
| 5. Nếu server thoát, các client đều phải nhận biết là ngắt kết nối. | 100% | Đạt đủ yêu cầu |
| 6. Tính năng chat riêng tư: Trong quá trình hội thảo, 1 client có thể chọn 1 client khác để chat riêng tư. Cửa sổ chat riêng tư sẽ khác cửa sổ hội thảo chung. | 90% | Chưa mở cửa sổ chat  riêng tư |
| 7. Client A có thể gửi file cho client B khi đang chat riêng tư. Tại 1 thời điểm chỉ 1 client gửi và chỉ gửi 1 file. | 100% | Đạt đủ yêu cầu |
| 8. Trong phòng chat chung, tại 1 thời điểm chỉ cho phép 1 client gửi file cùng lúc đến tất cả các client khác | 100% | Đạt đủ yêu cầu |
| 9. Xử lý đa luồng | 100% | Đạt đủ yêu cầu |
| 10. Giao diện | 10% | Chỉ có giao diện Console |
| **Toàn bộ project** | **95%** |  |

6. Nguồn tham khảo

**WinSock2:**

<https://docs.microsoft.com/en-us/windows/win32/api/winsock2/>

<https://www.geeksforgeeks.org/socket-programming-in-java/>

<https://www.geeksforgeeks.org/socket-programming-python/?ref=rp>

<https://docs.microsoft.com/en-us/windows/win32/winsock/complete-client-code>

<https://docs.microsoft.com/en-us/windows/win32/winsock/complete-server-code>

**Multithread:**

<https://www.geeksforgeeks.org/multithreading-in-java/>

<https://www.tutorialspoint.com/operating_system/os_multi_threading.htm>

<https://viblo.asia/p/lam-quen-voi-multithreading-trong-c-qm6RWQYXGeJE>

**FILE:**

<https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.socket.sendfile?view=netcore-3.1>

<https://stackoverflow.com/questions/15170161/c-winsock-sending-file>