ĐẠI HỌC QUỐC GIA HÀ NỘI **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ Độc lập – Tự do – Hạnh phúc**

**B Á O C Á O**

**BÀI TẬP LỚN môn Lập trình mạng**

**Lớp: INT3304 1**

**Phạm Việt Khanh 14020235**

**Nguyễn Khánh Hưng 14020224**

**Nguyễn Đức Huy 14020207**

*Hà Nội, 2017*

**Mục lục**

## 

[**1. Mục tiêu** 3](#_Toc501403122)

[**1.1 Tính năng chính:** 3](#_Toc501403123)

[**2. Môi trường chạy của chương trình** 3](#_Toc501403124)

[**3. Mô tả chương trình** 3](#_Toc501403125)

[**3.1 Thư viện cần có:** 3](#_Toc501403126)

[**3.2 Chạy chương trình:** 4](#_Toc501403127)

[**3.3 Mô tả code:** 4](#_Toc501403128)

[**3.4 Mô tả thiết kế chương trình:** 5](#_Toc501403129)

[**3.5 Mô tả giao thức:** 5](#_Toc501403130)

[**3.6 Quá trình thực hiện:** 6](#_Toc501403131)

[**3.7 Hình ảnh chạy chương trình** 7](#_Toc501403132)

[**4. Source code chương trình** 9](#_Toc501403133)

# **1. Mục tiêu**

Chương trình chat sử dụng giao thức kiểu publish/subscribe như MQTT bao gồm các thành phần

* Server đóng vai trò của MQTT Broker
* Client đóng vai trò vừa là Publisher (bên gửi thông tin) vừa là Subcriber

(bên nhận thông tin).

## **1.1 Tính năng chính:**

* Cho phép người dùng nói chuyện trực tiếp với nhau
* Cho phép người dùng nói chuyện theo nhóm
* Cho phép người dùng gửi file cho nhau
* Cho phép người dùng gửi file cho một nhóm

# **2. Môi trường chạy của chương trình**

* Hệ điều hành: Linux, macOS.
* Programming language: C.

# **3. Mô tả chương trình**

## **3.1 Thư viện cần có:**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <pthread.h>

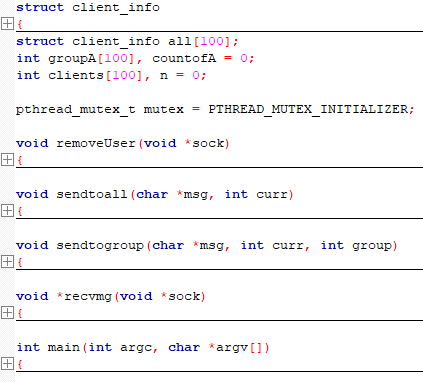
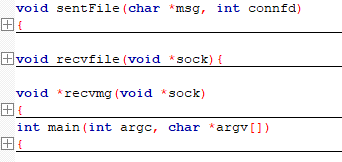
## **3.2 Chạy chương trình:**

* Server mở cổng 5555 (có thể thay) chờ kết nối TCP từ client. Cú pháp của lệnh chạy chương trình: **./server**
* Các client thiết lập kết nối với server qua cổng đã tạo. Cú pháp của lệnh chạy chương trình: **./client**
* Người dùng nhập tên đăng nhập
* Người dùng nhập chọn kiểu chat (chat all, group, 1vs1)
* 1. Chat all: Chat với tất cả client đang online
* 2. Chat group: Chat với các client trong nhóm được tạo sẵn.
* 3. Chat 1vs1: Chat với client với id tương ứng
* Trong lúc chat có thể gửi file, Client nhập cú pháp **FILE:filename.** Client upload file lên server, Client/group/all download file từ server bằng lệnh

**---f:filename**

* Server nhận thông báo online từ các client và hiển thị client connected
* Người dùng nhập dòng chat vào chương trình client, client gửi dòng chat cho server. Server gửi dòng chat cho các client khác
* Client nhập ký tự kết thúc @. Client đóng kết nối và kết thúc chương trình.

## **3.3 Mô tả code:**

**Client Server**

## **3.4 Mô tả thiết kế chương trình:**

**Void sentFile()**: Hàm đọc file local gửi lên server

**Void recvFile()**: Hàm nhận file server gửi đến

**Void\*recvmg()**: Nhận tin nhắn từ server.

**Struct client\_info**: Cấu trúc chứa thông tin client đã online

**Void removeUser()**: Xóa 1 client khi nó offline

**Void sendtoall()**: Gửi tin nhắn đến tất cả client online

**Void sendtogroup()**: Gửi tin nhắn đến group

**Void \*recvmg()**: Nhận tin nhắn từ client gửi lên rồi chuyển cho client khác

**Hàm main():** Chạy chương trình và sử dụng các hàm đã được viết sẵn để thực thi các tính năng cần thiết

**Hàm pthread\_mutex\_lock(&mutex):** Được sử dụng để tránh các client sử dụng chung một biến khi cùng kết nói đến 1 thời điểm.

## **3.5 Mô tả giao thức:**

****Chương trình chat sử dụng giao thức kiểu publish/subscribe như MQTT bao gồm các thành phần.

**Server đóng vai trò của MQTT Broker**

* Nhận tất cả thông báo từ clients, lọc(\*) chúng và gửi thông báo cho tất cả các client/group/all
* Xử lý việc chứng thực/chứng nhận giữa các clients

**Client đóng vai trò vừa là Publisher (bên gửi thông tin) vừa là Subcriber (bên nhận thông tin).**

* Một bên gửi (publisher), một hoặc nhiều bên nhận (subscribers)
* Sử dụng Key để phân biệt bên nhận thông tin.
* char key[ ] = "111"; // loai chat all
* char key1[ ] = "100"; // loai tin nhan GroupA
* char key2[ ] = "102"; //loai tin nhan Group B

## **3.6 Quá trình thực hiện:**

Khi User nhập tên, bên client sẽ dùng hàm write gửi tên lên server và server lưu vào trong mảng ***all[100]*** với cấu trúc ***client\_info***gồm cổng kết nối, tên, id group.

Lúc này server và clients bắt đầu chạy hàm **recvmg** được tạo trong luồng và thực hiện thao tác theo yêu cầu với **key gửi đi và nhận đến** tương ứng.

Menu hiện ra gồm 3 tùy chọn như trên mô tả chương trình.

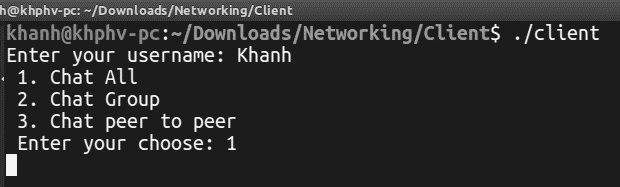
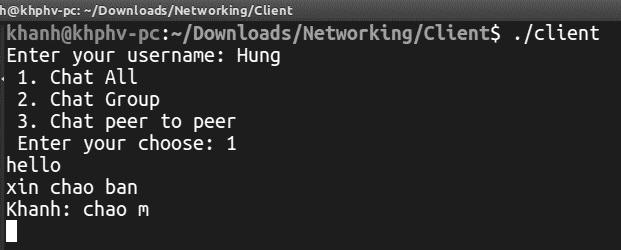
* Nếu chọn 2 - 1: Chat group A, Client gửi mess có nối key gửi đến là 100, hàm **recvmg** chạy đến vòng if đầu tiên 🡪 đúng, tính năng chat Group A được chạy trong hàm. Bỏ đuôi key và gửi lại bằng hàm **sentogroup.**
* Nếu chọn 2 - 2: Chat group B, Client gửi mess có nối key gửi đến là 102, hàm **recvmg** chạy đến vòng if thứ 2 🡪 đúng, tính năng chat Group B được chạy trong hàm. Bỏ đuôi key và gửi lại bằng hàm **sentogroup**.
* Nếu chọn 3: Chat 1 vs 1, Client gửi mess có nối key gửi đến là 103, hàm **recvmg** chạy đến vòng if thứ 3 🡪 đúng, tính năng chat 1 vs 1 được chạy trong hàm. Bỏ đuôi key và gửi lại bằng hàm **sentogroup, (Group ở đây có 2 clients)**
* Nếu chọn 1: Chat all, Client gửi mess có nối key gửi đến là 111, hàm **recvmg** chạy đến trường hợp cuối, tính năng chat all được chạy trong hàm. Bỏ đuôi key và gửi lại bằng hàm **sentoall**.

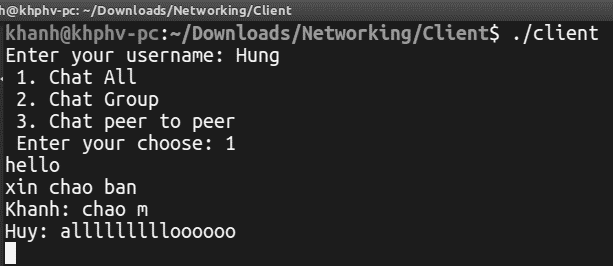
Upload/Download file: Khi client nhập FILE:filename, vòng if gặp điều kiện đúng và chạy hàm **sentFile()** rồi tiếp tục. File được gửi lên server. Client download với cú pháp ---f:filename và thực hiện hàm **recvFile()**­, tên file sẽ được gửi lên server, server kiểm tra và thông báo đồng thời gửi cho client file cần download.

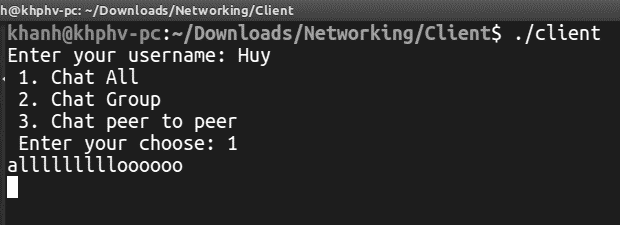
Trong khi nhập liệu từ client, nếu ký tự là @ thì chương trình client thoát.

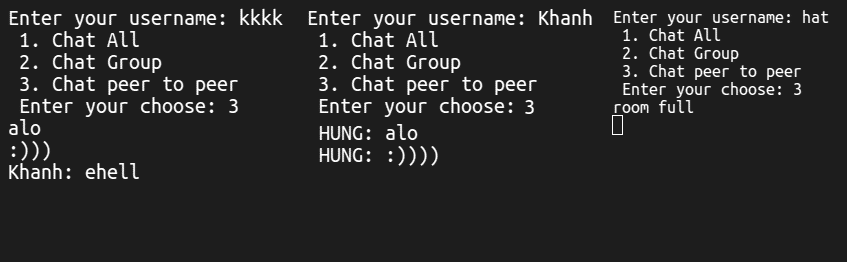
## **3.7 Hình ảnh chạy chương trình**

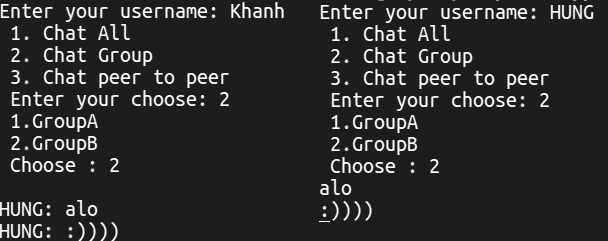
Chat all:



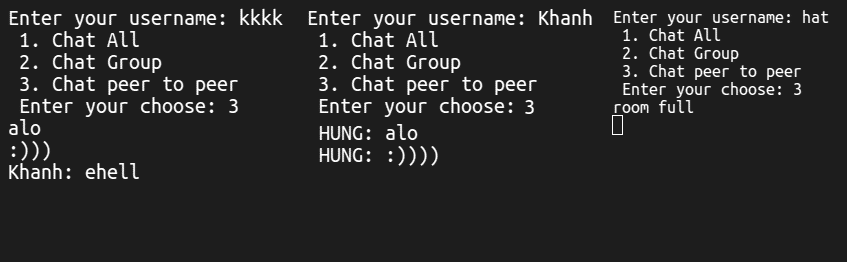




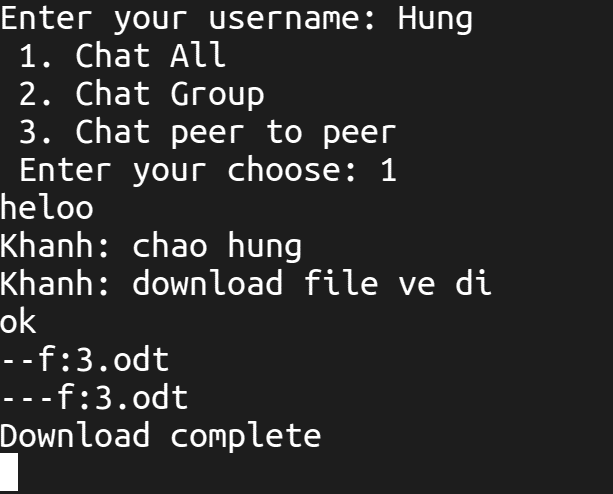
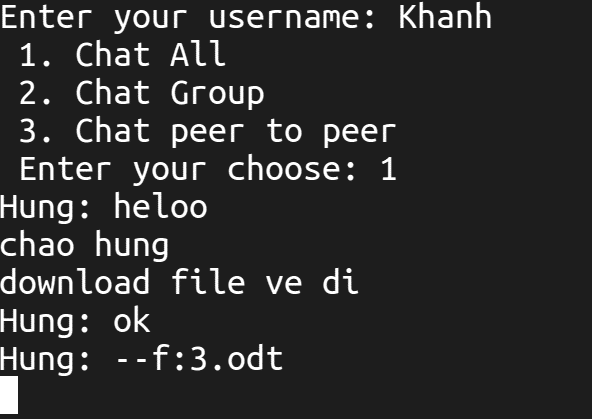
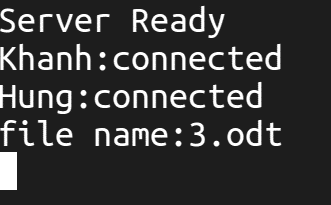
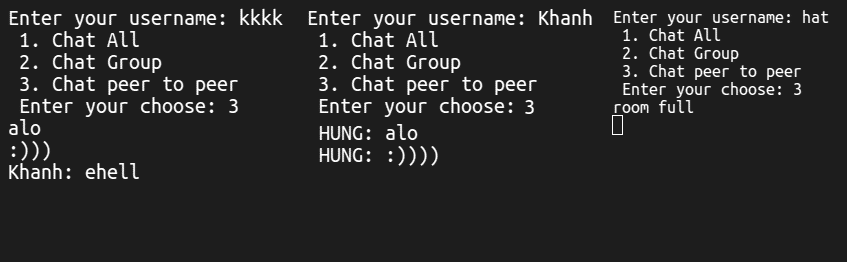
Chat group:

****

Chat 1 vs 1



Gửi file



# **4. Source code chương trình**

Link github, có ghi log trong quá trình sửa lỗi của từng thành viên trong nhóm. [*https://github.com/Khanhphv/Net*](https://github.com/Khanhphv/Net)

----------- Hết -----------