

**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**CƠ SỞ NGÀNH MẠNG**

**NGUYÊN LÝ HỆ ĐIỀU HÀNH: XÂY DỰNG CHƯƠNG TRÌNH GAME CÓ SỬ DỤNG THUẬT TOÁN TRÁNH DEADLOCK**

**LẬP TRÌNH MẠNG: XÂY DỰNG ỨNG DỤNG TRUYỀN FILE ĐƠN GIẢN SỬ DỤNG GIAO THỨC FTP**

*Giảng viên hướng dẫn: ThS. Nguyễn Thị Lệ Quyên*

*Nhóm: 17.14A*

*Sinh viên thực hiện: Trần Nhàn*

*Lớp: 17TCLC1*

*Mã số sinh viên: 102170237*

***Đà Nẵng, 12/2020***

MỤC LỤC

[PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH 4](#_Toc59923777)

[TIÊU ĐỀ: XẤY DỰNG CHƯƠNG TRÌNH GAME CÓ SỬ DỤNG THUẬT TOÁN TRÁNH DEADLOCK. 4](#_Toc59923778)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 4](#_Toc59923779)

[1.1 Tiến trình 4](#_Toc59923780)

[1.1.1 Khái niệm 4](#_Toc59923781)

[1.1.2 Các trạng thái của tiến trình 4](#_Toc59923782)

[1.2. Deadlock 5](#_Toc59923783)

[1.2.1 Khái niệm 5](#_Toc59923784)

[1.2.2. Điều kiện xảy ra Deadlock 5](#_Toc59923785)

[1.2.3. Ngăn chặn Deadlock 6](#_Toc59923786)

[1.3 Thuật toán Banker 7](#_Toc59923787)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 8](#_Toc59923788)

[2.1 Ý tưởng 8](#_Toc59923789)

[2.2 Phân tích các chức năng 8](#_Toc59923790)

[2.2.1 Các chức năng của client 8](#_Toc59923791)

[2.2.2 Các chức năng của server 8](#_Toc59923792)

[2.3 Xây dựng chương trình 9](#_Toc59923793)

[2.3.1 Class Client: 9](#_Toc59923794)

[2.3.2 Class Server 9](#_Toc59923795)

[2.3.3 Class Game 10](#_Toc59923796)

[2.3.4 Trao đổi giữa người chơi(client) và hệ thống(server) 12](#_Toc59923797)

[CHƯƠNG 3. DEMO CHƯƠNG TRÌNH 13](#_Toc59923798)

[CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 18](#_Toc59923799)

[3.1 Kết quả đạt được 18](#_Toc59923800)

[3.2 Vấn đề chưa giải quyết 18](#_Toc59923801)

[3.3 Hướng phát triển 18](#_Toc59923802)

[PHẦN II: LẬP TRÌNH MẠNG 19](#_Toc59923803)

[TIÊU ĐỀ: TÌM HIỂU VỀ GIAO THỨC FTP VÀ XÂY DỰNG CHƯƠNG TRÌNH TRUYỀN FILE ĐƠN GIẢN CÓ SỬ DỤNG FTP. 19](#_Toc59923804)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 19](#_Toc59923805)

[1.1 Giao tiếp Socket và mô hình Client-Server 19](#_Toc59923806)

[1.1.1 Socket 19](#_Toc59923807)

[1.1.2 Mô hình client – server 20](#_Toc59923808)

[2.2 Giao thức FTP 20](#_Toc59923809)

[2.2.1 Khái niệm 20](#_Toc59923810)

[2.2.2 So sánh FTP và FTPS 20](#_Toc59923811)

[2.2.3 Cơ chế hoạt động của FTP 21](#_Toc59923812)

[2.2.4 Mô hình hoạt động của FTP 21](#_Toc59923813)

[2.2.5 Trình tự truy cập và chứng thực FTP 23](#_Toc59923814)

[CHƯƠNG 2. XÁY DỰNG CHƯƠNG TRÌNH VÀ DEMO 24](#_Toc59923815)

[3.1 Các thành phần chính sử dụng trong chương trình 24](#_Toc59923816)

[CHƯƠNG 3. KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN 27](#_Toc59923817)

[3.1 Kết quả đạt được 27](#_Toc59923818)

[3.2 Vấn đề chưa giải quyết được 27](#_Toc59923819)

[3.3 Hướng phát triển 27](#_Toc59923820)

# PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

# TIÊU ĐỀ: XẤY DỰNG CHƯƠNG TRÌNH GAME CÓ SỬ DỤNG THUẬT TOÁN TRÁNH DEADLOCK.

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### 1.1 Tiến trình

#### 1.1.1 Khái niệm

- Tiến trình(process) là một chương trình đang xử lý, sỡ hữu một con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành tác vụ của mình, một tiến trình có thể cần đến một số tài nguyên như CPU, bộ nhớ chính, các tập tin, và thiết bị nhập/xuất.

#### 1.1.2 Các trạng thái của tiến trình

- Trạng thái của tiến trình tại một thời điểm được xác định bởi hoạt động hiện thời của tiến trình tại thời điểm đó. Trong quá trình sống một tiến trình thay đỏi trạng thái do nhiều nguyên nhân như: phải chờ một sự kiện nào đó xảy ra, hay đợi một tác động nhập/xuất hoàn tất, buộc phải dừng thời gian hoạt động để xử lý…

- Tại một thời điểm, một tiến trình có thể nhận một trong các trạng thái sau:

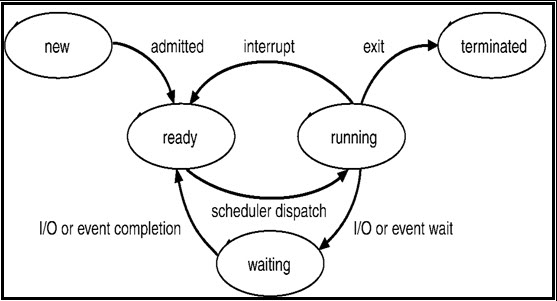
\* Mới tạo: tiến trình đang được tạo lập.

\* Running: các chỉ thị của tiến trình đang được xử lý.

\* Blocked: tiến trình chờ được cấp phát một tài nguyên, hay chờ một sự kiện xảy ra.

\* Ready: tiến trình chờ được cấp phát CPU đẻ xử lý.

\* Kết thúc: tiến trình hoàn tất xử lý.



Hình 1.Sơ đồ chuyển trạng thái của một tiến trình.

### 1.2. Deadlock

#### 1.2.1 Khái niệm

- Deadlock là trạng thái khi mỗi tiến trình trong tập hợp các tiến trình đều chờ một sự kiện mà chỉ có một tiến trình khác trong tập hợp mới có thể phát sinh được mà tiến trình đó cũng đang ở trạng thái block chiếm giữ. Như vậy không có tiến trình nào có thể tiếp tục xử lý, cũng như giả phóng tài nguyên cho tiến trình khác sử dụng, tất cả các tiến trình trong tập hợp điều bị khóa vĩnh viễn. Người ta gọi tình trạng đó là deadlock.

#### 1.2.2. Điều kiện xảy ra Deadlock

- Có sử dụng tài nguyên không chia sẻ(Mutual exclusion): Mỗi tài nguyên không thể sử dụng bởi nhiều hơn một tiến trình tại một thời điểm.

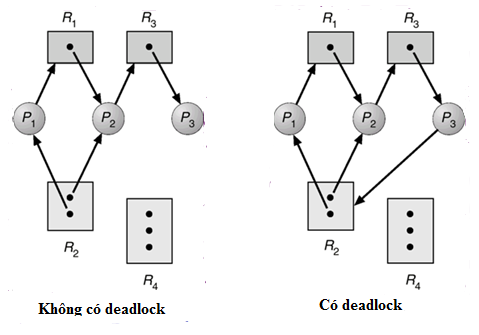
- Sự chiếm giữ và yêu cầu tài nguyên(Wait for): Các tiến trình giữ tài nguyên đã được cấp phát cho nó và chờ được cấp phát thêm tài nguyên mới.

- Không thu hồi tài nguyên từ tiến trình đang giữ chúng(No preemption): Các tài nguyên không thể bị đòi lại, chúng chỉ có thể được giải phóng bởi chính tiến trình chiếm giữ chúng.

- Tồn tại một chu kỳ trong đồ thị cấp phát tài nguyên(Circular wait): Các tiến trình giử tài nguyên và chờ các tài nguyên bị giử bởi tiến trình khác, tạo thành một chu trình.

\* Ví dụ: Tiến trình P1, chiếm R1, chờ R2. Tiến trình P2 chiếm R2, chờ R3,... Tiến trình N chiếm Rn, chờ R1.

- Khi có đủ 4 điều kiện này thì Deadlock xảy ra. Nếu thiếu 1 trong 4 điều kiện trên thì không xảy ra Deadlock.



Hình 2.Sơ đồ trạng thái deadlock.

#### 1.2.3. Ngăn chặn Deadlock

- Để deadlock không xảy ra, cần đảm bảo 1 trong 4 điều kiện cần không xảy ra:

\* Tài nguyên không thể chia sẻ: hầu như không thể tránh được điều kiện này vì bản chất tài nguyên gần như cố định. Tuy nhiên đối với một số tài nguyên về kết xuất, người ta có thể dùng các cơ chế spooling để biến đổi thành tài nguyên có thể chia sẻ.

\* Sự chiếm giữ và yêu cầu thêm tài nguyên: phải bảo đảm rằng mỗi khi tiến trình yêu cầu thêm một tài nguyên thì nó không chiếm giữ các tài nguyên khác. Có thể áp dặt một trong hai cơ chế truy xuất sau :

+ Tiến trình phải yêu cầu tất cả các tài nguyên cần thiết truớc khi bắt dầu xử lý => phương pháp này có khó khăn là tiến trình khó có thể uớc luợng chính xác tài nguyên cần sử dụng vì có thể nhu cầu phụ thuộc vào quá trình xử lý . Ngoài ra nếu tiến trình chiếm giữ sẵn các tài nguyên chưa cần sử dụng ngay thì việc sử dụng tài nguyên sẽ kém hiệu quả.

+ Khi tiến trình yêu cầu một tài nguyên mới và bị từ chối, nó phải giải phóng các tài nguyên đang chiếm giữ , sau đó lại đuợc cấp phát trở lại cùng lần với tài nguyên mới.

=> phương pháp này làm phát sinh các khó khăn trong việc bảo vệ tính toàn vẹn dữ liệu của hệ thống.

\* Không thu hồi tài nguyên: cho phép hệ thống duợc thu hồi tài nguyên từ các tiến trình bị khoá và cấp phát trở lại cho tiến trình khi nó thoát khỏi tình trạng bị khóa. Tuy nhiênvới một số loại tài nguyên, việc thu hồi sẽ rất khó khan vì vi phạm sự toàn vẹn dữ liệu .

\* Tồn tại một chu kỳ: tránh tạo chu kỳ trong dồ thị bằng cách cấp phát tài nguyên theo một sự phân cấp nhu sau :

gọi R = {R1, R2,...,Rm} là tập các loại tài nguyên.

- Các loại tài nguyên duợc phân cấp từ 1-N.

Ví dụ : F(dia) = 2, F(máy in) = 12.

- Các tiến trình khi yêu cầu tài nguyên phải tuân thủ quy dịnh : khi tiến trình dang chiếm giữ tài nguyên Ri thì chỉ có thể yêu cầu các tài nguyên Rj nếu F(Rj) > F(Ri).

### 1.3 Thuật toán Banker

- Giải thuật đồ thị cấp phát tài nguyên không thể áp dụng tới hệ thống cấp phát tài nguyên với nhiều thể hiện của mỗi loại tài nguyên. Giải thuật tránh deadlock mà chúng ta mô tả tiếp theo có thể áp dụng tới một hệ thống nhưng ít hiệu quả hơn cơ chế đồ thị cấp phát tài nguyên. Giải thuật này thường được gọi là giải thuật của Banker.

- Tên được chọn vì giải thuật này có thể được dùng trong hệ thống ngân hàng để đảm bảo ngân hàng không bao giờ cấp phát tiền mặt đang có của nó khi nó không thể thoả mãn các yêu cầu của tất cả khách hàng. Khi một quá trình mới đưa vào hệ thống, nó phải khai báo số tối đa các thể hiệncủa mỗi loại tài nguyên mà nó cần. Số này có thể không vượt quá tổng số tài nguyên trong hệ thống. Khi một người dùng yêu cầu tập hợp các tài nguyên, hệ thống phải xác định việc cấp phát của các tài nguyên này sẽ để lại hệ thống ở trạng thái an toàn hay không. Nếu trạng thái hệ thống sẽ là an toàn, tài nguyên sẽ được cấp; ngược lại quá trình phải chờ cho tới khi một vài quá trình giải phóng đủ tài nguyên.

## CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

### 2.1 Ý tưởng

Ta sẽ tạo ra một trò chơi nấu ăn, với luật lệ như sau.

* Trò chơi sẽ có 2 người chơi. Mỗi người chơi cần hoàn thành 3 món ăn và nộp lên hệ thống để thắng cuộc.
* Để hoàn thành một món ăn, người chơi phải tập hợp đủ các công cụ dùng để nấu món ăn đó.
* Sau khi hoàn thành một món ăn người chơi phải trả lại tất cả công cụ đang giữ.
* Để có được các công cụ nấu ăn, người chơi phải gửi yêu cầu lên hệ thống. Hệ thống sẽ quyết định có cung cấp tài nguyên cho người chơi hay không.

Trong quá trình chơi mỗi khi người chơi yêu cầu một công cụ nào đó, hệ thống sẽ kiểm tra xem nếu cấp công cụ đó cho người chơi thì sẽ gây ra deadlock hay không, nếu có hệ thống sẽ từ chối cấp công cụ nấu ăn cho người chơi.

Hệ thống sẽ dùng thuật toán banker để phát hiện deadlock, vì thế người chơi bắt buộc phải chọn món ăn để nấu trước khi yêu cầu tài nguyên

### 2.2 Phân tích các chức năng

Ta sẽ xây dựng một game theo mô hình client - server.

#### 2.2.1 Các chức năng của client

Chương trình client của người chơi sẽ có những chức năng sau:

1. Gửi yêu cầu kết nối đến server: người chơi sẽ nhập tên server rồi kết nối đến server và chờ đến khi có đủ 2 người chơi sẽ bắt đầu game.
2. Thay đổi món ăn hiện tại của mà người chơi đang nấu: khi game bắt đầu server sẽ mặc định chọn sẵn cho người chơi một món ăn để nấu, nếu người chơi không muốn có thể chọn món khác.
3. Yêu cầu công cụ để nấu ăn: để hoàn thành món ăn người chơi sẽ gửi yêu cầu đến hệ thống để yêu cầu lấy công cụ nấu ăn.
4. Nấu và nộp món ăn lên hệ thống: sau khi có đủ các công cụ cần thiết người chơi sẽ nấu và nộp món ăn đó lên cho hệ thống để hoàn thành món ăn và đổi sang món ăn khác

#### 2.2.2 Các chức năng của server

Server sẽ có các chức năng sau:

1. Kết nối và quản lý các client, kiểm tra số lượng người chơi đang kết nối đến và bắt đầu khi đủ người. Kiểm tra nếu có người chơi thoát giữ chừng thì thông báo game kết thúc.
2. Xử lý những gì liên quan đến logic game, như lưu trữ trạng thái trò chơi, nhận và xử lý yêu cầu của người chơi.

### 2.3 Xây dựng chương trình

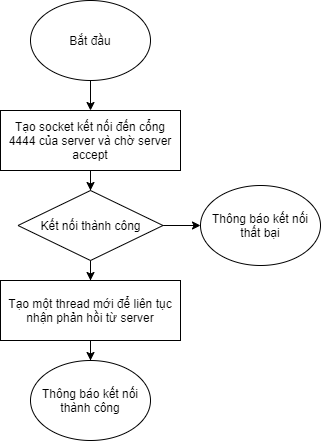
Ta sẽ xây dựng chương trình gồm các module sau:

* Ở phía client sẽ có class Client đảm nhiệm xử lý logic và class, ClientUI đảm nhiệm việc hiển thị giao diện.
* Ở phía server sẽ có các class Server dảm nhiệm việc quản lý các client và class Game để xử lý logic game.

#### 2.3.1 Class Client:

Class client sẽ có nhưng phương thức chính như sau:

* connect(): kết nối đến server và chờ bắt đầu game.



Hình 3:sơ đồ khối của hàm conncet

* changeFood: Client sẽ gửi gói tin chứa yêu cầu thay đổi món ăn hiện tại và server sẽ gửi lại gói tin chứa trạng thái sau khi thay đổi
* requestTool: Client sẽ gửi gói tin chứa yêu cầu công cụ nấu ăn và server sẽ gửi về trạng thái của người chơi sau khi đã xử lý(có thay đổi hoặc không).
* submitFood: Client sẽ gửi gói tin chứa yêu cầu nộp món ăn lên và server sẽ kiểm tra việc nộp có thành công hay không, và người chơi đó có thắng cuộc hay không sau đó trả về kết quả cho Client.

#### 2.3.2 Class Server

Class server sẽ có những thuộc tính sau:

* ServerSocket server: server socket dùng để chấp nhận kết nối đến các client.
* int port: có giá trị mặc định là 4444, là giá cổng mặc định khi thực hiện client kết nối đến server.
* List<Thread> clients: vói mỗi client server sẽ tạo ra một luồng xử lý riêng biệt để quản lý việc gửi và nhận.
* Game game: là một đối tượng của class game để quản lý trạng thái trò chơi, sẽ được khởi tạo mỗi khi bắt đàu một game mới.

Các phương thức chính của class Server:

* start: khởi tạo server socket và bắt đầu chấp nhận kết nối từ client
* stop: dừng và kết thuc chương trình khi thỏa mnx một số điều kiện
* removeAllClient: thông báo kết quả trò chơi và disconnect tất cả các client đang kết nối đến server.
* getCommandAndExec: nhận lấy request từ client và gọi đến đối tượng game để xử lý logic game.

#### 2.3.3 Class Game

Class Game sẽ đảm nhận việc xử lý logic chính của game, và cung cấp các phương thức để server có thể gọi và nhận về kết quả.

Class Game sẽ có những thuộc tính sau:

1. Các trạng thái của người chơi

* currentFood: là một mảng để lưu trữ id của những món ăn mà người chơi đang nấu.
* completeState: một ma trận để lưu trữ trạng thái hoàn thành của mỗi món ăn của mỗi người chơi.
* available: mảng lưu số lượng từng công cụ mà hệ thống hiện có.
* allocated: ma trận lưu trữ số lượng từng công cụ mà mỗi người chơi đang giữ.
* maxNeed: ma trận lưu trữ số lượng tối đa cho mỗi công cụ mà người chơi cần để hoàn thành món ăn hiện tại.

1. Các dữ liệu về món ăn và công cụ:

* resourceNeedOfFood: ma trận lưu trữ công cụ mà mỗi món ăn cần.
* availResource: mảng lưu số lượng sẵn có của từng công cụ.

Các phương thức của class Game

1. INIT

Phương thức init được dùng để khởi tạo trạng thái ban dàu của trò chơi. Bao gồm việc khởi tạo giá trị cho các thuộc tính currentFood, completeState, available, allocated, maxNeed.

1. CHANGE\_FOOD

Phương thức CHANGE\_FOOD được xử dụng để thay đổi món ăn mà người chơi có id là playerId dang chọn thành món ăn có id là foodId, đồng thời lấy lại tất cả những công cụ mà người chơi đang giữ.

for(int i=0; i<numOfTool; i++) {

available[i]+=allocated[playerID][i];

allocated[playerID][i] = 0;

}

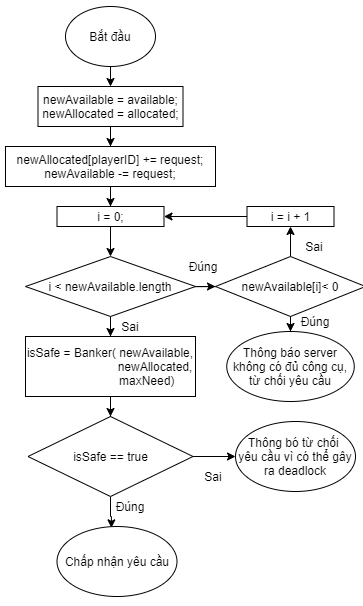
curentFood[playerID] = foodID;

maxNeed[playerID] = foodResourceNeed[foodID];

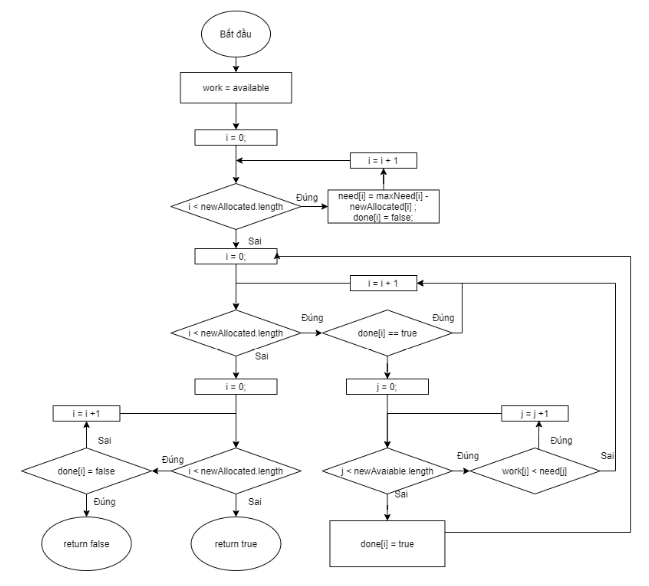
setReturnMessage("success to change food");

1. REQUEST\_TOOL

Phương thức REQUEST\_TOOL sẽ trả về việc cấp phát công cụ có thể thực hiện được hy không



Hình 4. Sơ đồ khối của hàm REQUEST\_TOOL



Hình 5. Sơ đồ khối của hàm Banker

1. Phương thức SUBMIT\_FOOD

Phương thức SUBMIT\_FOOD sẽ kiểm tra người chơi có thể hoàn thành món ăn hay không, nếu có thì sẽ update trạng thái người chơi đồng thời kiểm tra xem người chơi đó có thắng cuộc hay chưa

#### 2.3.4 Trao đổi giữa người chơi(client) và hệ thống(server)

Người chơi(client) và hệ thống(server) sẽ gửi cho nhau những gói tin bằng giao thức TCP cho nhau và tùy vào từng gói tin thì sẽ có các cách xử lý khác nhau.

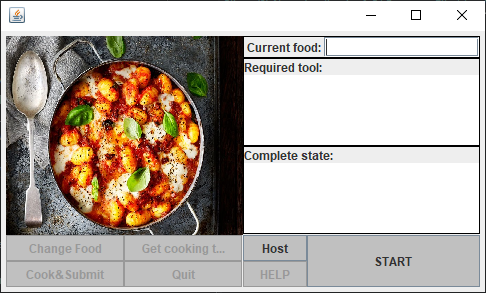
Những gói tin mà server gửi cho client:

* Connected: thông báo người chơi đã kết nối đến server.
* Data: bao gồm thông tin về các món ăn và các công cụ.
* Ready: thông báo trò chơi đã đủ người và chuẩn bị bắt đầu.
* State: chứa trạng thái của người chơi.
* Finish: thông báo trò chơi đã kết thúc và người chiến thắng.

Những gói tin mà client gửi cho server:

* Change\_food: yêu cầu đến hệ thống để thay đổi món ăn hiện đang nấu.
* Request\_for\_resource: yêu cầu hệ thống cung cấp các công cụ nấu ăn.
* Send\_food: gửi món ăn đã hoàn thành lên cho hệ thống.

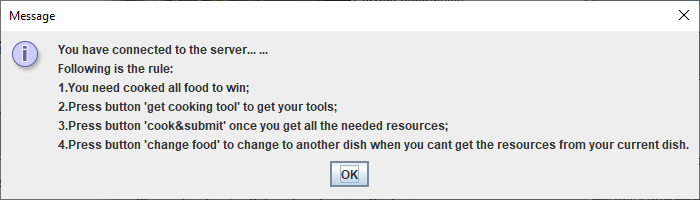
## CHƯƠNG 3. DEMO CHƯƠNG TRÌNH



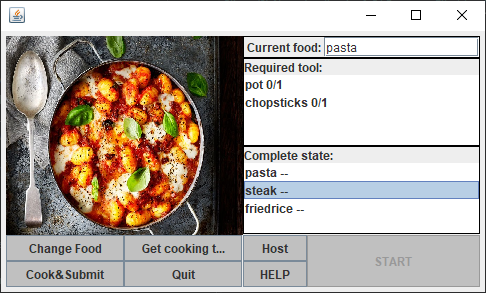
Hình 6 : Giao diện của client khi chưa kết nối đến server

Giao diện của client sẽ gồm có:

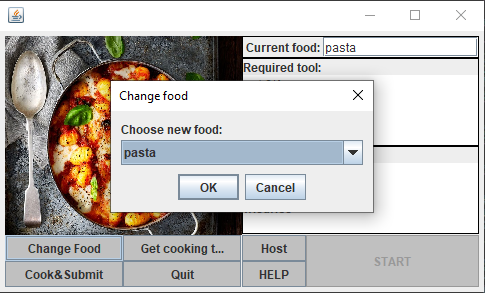
* Nút START: Nhấn vào sẻ kết nối đến server
* Nút Host: chọn server sẽ kết nối đến, mặc định là localhost
* Nút HELP: Nhấn vào sẽ hiển thị luật chơi
* Nút Change Food: Nhấn vào để chọn món ăn khác.
* Nút Get cooking tool: Nhấn vào để yêu cầu lấy công cụ nấu ăn
* Nút Cook&Submit để nộp món ăn khi đã có đủ công cụ
* Curent Food: tên của món ăn đang nú hiện tại
* Required tool: công cụ cần thiết để nấu món ăn hiện tại



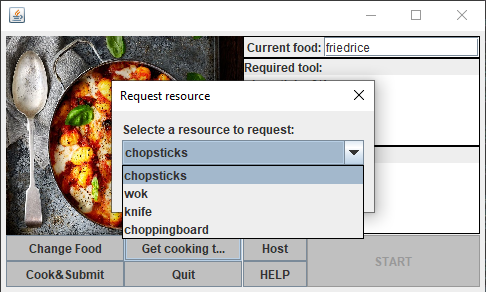
Hình 7 : Nhấn vào button HELP để hiển thị luật chơi



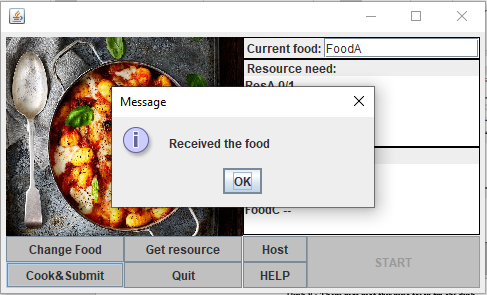
Hình 8 : Sau khi kết nối đến server, client sẽ hiển thị trạng thái hoàn thành của các món ăn và tài nguyền cần thiết cho món ăn hiện tại



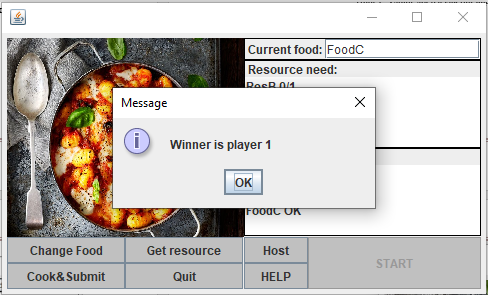
Hình 9 : Khi người chơi nhấn vào nút Change food chọn một món ăn mới



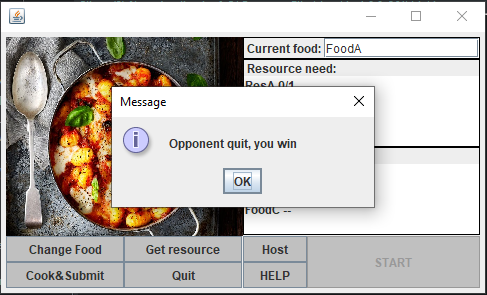
Hình 10 : khi người chơi nhấn vào nút Get resource để yêu cầu một tài nguyên



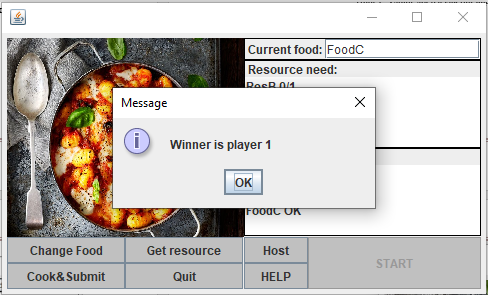
Hình 11 : Sau khi người chơi có đủ các tài nguyên và nhấn nút submit



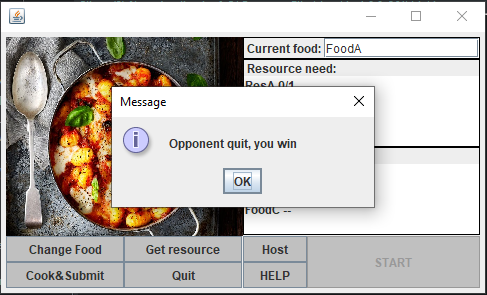
Hình 12 : Khi người chơi nấu đủ các món ăn thì sẽ thông báo kết thúc game đến tất cả người chơi



Hình 13: Khi có người chơi thoát khỏi sẽ thông báo đến người chơi còn lại



Hình 14 : Khi người chơi nấu đủ các món ăn thì sẽ thông báo kết thúc game đến tất cả người chơi



Hình 15: Khi có người chơi thoát khỏi sẽ thông báo đến người chơi còn lại

## CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 3.1 Kết quả đạt được

* Thông qua đồ án này em đã được tìm hiểu kỹ hơn về deadlock, nguyên nhân gây ra và cách xử lý.
* Chương hệ thống giải quyết được vấn đề cần phải cung cấp tài nguyên một cách hợp lý mà không tránh gây ra tình trạng deadlock xảy ra.
* Chương trình được xây dựng theo mô hình client – server. Chương trình thể hiện kỹ thuật lập trình với socket theo bộ giao thức TCP/IP. Có thể xử lý các tác vụ song song giúp cho người dùng không phải chờ cho đến khi yêu cầu được hoàn thành mới có thể làm tiếp.

- Chương trình hoàn thành cơ bản mục tiêu được đề ra, giao diện đơn giản, dễ sử dụng.

### 3.2 Vấn đề chưa giải quyết

* Hệ thống hiện chỉ có thể chơi với 2 người chơi.
* Hệ thống mỗi lần chơi chỉ có thể chơi với hai người chơi.

### 3.3 Hướng phát triển

* Cho phép nhiều người có thể chơi cùng một game
* Cho phép client có thể tạo nhiều phòng chơi và các client có thể mời nhau để chơi chung

# PHẦN II: LẬP TRÌNH MẠNG

# TIÊU ĐỀ: TÌM HIỂU VỀ GIAO THỨC FTP VÀ XÂY DỰNG CHƯƠNG TRÌNH TRUYỀN FILE ĐƠN GIẢN CÓ SỬ DỤNG FTP.

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### 1.1 Giao tiếp Socket và mô hình Client-Server

#### 1.1.1 Socket

##### 1.1.1.1. Socket là gì?

* Socket là một điểm cuối (end-point) của liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 process nói chuyện với 1 process khác.
* Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

##### 1.1.1.2 Nguyên lý hoạt động của socket

* Chức năng của socket là kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận giữ liệu qua Internet. Giao diện lập trình ứng dụng mạng này chỉ có thể hoạt động khi đã có thông tin về thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau.
* Để 2 ứng dụng cần truyền thông tin phải đáp ứng điều kiện sau thì socket mới có thể hoạt động:
* 2 ứng dụng có thể nằm cùng trên một máy hoặc 2 máy khác nhau
* Trong trường hợp 2 ứng dụng cùng nằm trên một máy, số hiệu cổng không được trùng nhau.

##### 1.1.1.3 Phân loại socket

* Socket được chia làm 4 phân loại khác nhau. Dưới đây, hãy cùng Mắt Bão tìm hiểu chi tiết về:
* Stream Socket: là socket hoạt động thông qua giao thức TCP (Transmission Control Protocol). Stream Socket chỉ hoạt động khi server và client đã kết nối với nhau.
* Datagram Socket: là socket hoạt động thông qua giao thức UDP ( User Datagram Protocol). Datagram Socket có thể hoạt động kể cả khi không có sự thiết lập kết nối giữa 2 máy với nhau.
* Websocket: là công cụ hỗ trợ việc kết nối qua lại trên internet giữa client và server. Giúp diễn ra nhanh chóng và hiệu quả hơn thông qua việc sử dụng TCP socket.

Unix socket:  là điểm giao tiếp hỗ trợ trao đổi giữa các ứng dụng khác nhau ngay trên cùng máy tính. Mọi hoạt động Unix socket diễn ra ngay ở nhân của hệ điều hành.

#### 1.1.2 Mô hình client – server

##### 1.1.2.1 Khái niệm về mô hình client – server

* Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.

##### 1.1.2.2 Chế độ hoạt động

* Trong mô hình này, server luôn hoạt động (chế độ luôn mở) để chờ phục vụ client. Chương trình server phải hoạt động trên máy có địa chỉ IP cố định.
* Client có thể hoạt đọng ở chế độ luốn mở hoặc mở không thường xuyên. Các client không giao tiếp trực tiếp với nhau mà chỉ giao tiếp với server. Chương trình client phải biết được địa chỉ của máy chủ.
* Do chương trình server luôn hoạt động và chạy trên máy tính có IP cố định và được công bố, chương trình khách có thể liên lạc với máy chủ bất kỳ khi nào cần sử dụng dịch vụ do chương trình máy chủ cung cấp.

##### 1.1.2.3. Giao tiếp client – server

* Việc giao tiếp giữa client và server được thực hiện dưới hình thức trao đổi các thông điệp (message):
* Client gửi thông điệp yêu cầu (request message) cho server để mô tả công việc cần thực hiện.
* Khi nhận được thông điệp này, server sẽ phân tích và xác định công việc phải thực hiện.
* Nếu quá trình xử lý sinh ra kết quả cần trả cho client thì server sẽ gửi cho client thông điệp trả lời (reply message).
* Định dạng và ý nghĩa của các thông điệp trao đổi giữa client và server được quy định bởi giao thức dùng trong ứng dụng

### 2.2 Giao thức FTP

#### 2.2.1 Khái niệm

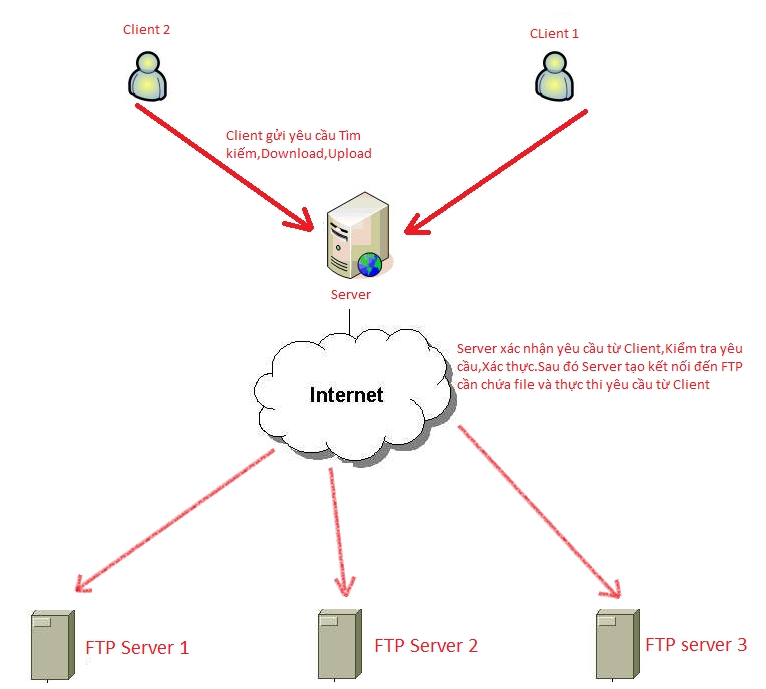
FTP là chữ viết tắc của File Transfer Protocol - Giao thức truyền file. FTP là một giao thức truyền file trên mạng dựa trên chuẩn TCP nên rất đáng tin cậy.

#### 2.2.2 So sánh FTP và FTPS

FTP là một giao thức không được bảo mật, nó truyền dữ liệu mà không thực hiện mã hóa vì vậy mà người dùng có thể gặp phải những rủi ro khi sử dụng giao thức này.

Một phương pháp hữu hiệu hơn đó là sử dụng giao thức FTPS (FTP trên SSL) cung cấp khả năng mã hóa giao thức cho các dữ liệu được truyền.

#### 2.2.3 Cơ chế hoạt động của FTP

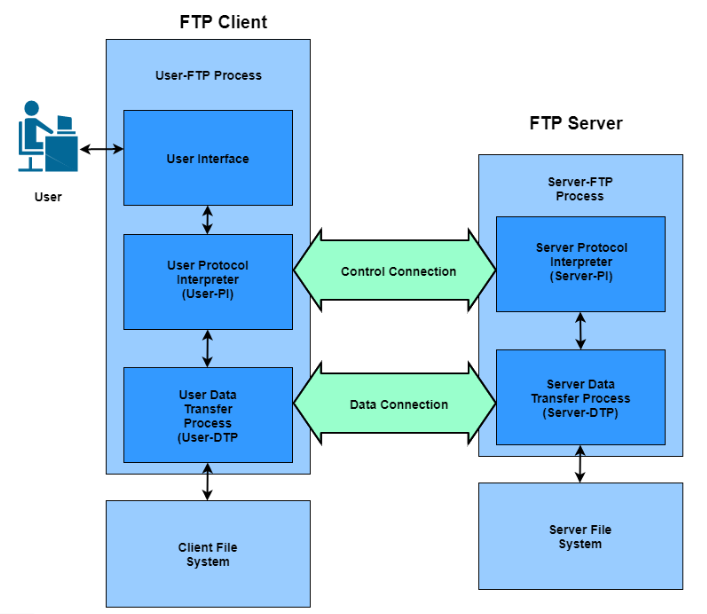
[](http://code.google.com/)

Hình 16 : Cơ chế hoạt động của FTP

#### 2.2.4 Mô hình hoạt động của FTP

##### 2.2.4.1 Tiến trình Server-FTP và User-FTP

FTP là một giao thức dạng client/server truyền thống, tuy nhiên thuật ngữ client thông thường được thay thế bằng thuật ngữ user – người dùng – do thực tế là người sử dụng mới là đối tượng trực tiếp thao tác các lệnh FTP trên các máy client. Bộ phần mềm FTP được cài đặt trên một thiết bị được gọi là một tiến trình. Phần mềm FTP được cài đặt trên máy Server được gọi là tiến trình Server-FTP, và phần trên máy client được gọi là tiến trình User-FTP.



Hình 17 : Đối chiếu các tiến trình vào trong mô hình FTP

Do các chức năng điều khiển và dữ liệu sử dụng các kênh khác nhau, nên mô hình hoạt động của FTP cũng chia phần mềm trên mỗi thiết bị ra làm hai thành phần logic tương ứng với mỗi kênh. Thành phần Protocol Interpreter (PI) là thành phần quản lý kênh điều khiển, với chức năng phát và nhận lệnh. Thành phần Data Transfer Process (DTP) có chức năng gửi và nhận dữ liệu giữa phía client với server. Ngoài ra, cung cấp cho tiến trình bên phía người dùng còn có thêm thành phần thứ ba là giao diện người dùng FTP - thành phần này không có ở phía server.

Do đó, có hai tiến trình xảy ra ở phía server, và ba tiến trình ở phía client. Các tiến trình này được gắn với mô hình FTP để mô tả chi tiết hoạt động của giao thức FTP.

##### 2.2.4.2 Các tiến trình phía server

Các tiến trình phía server bao gồm hai giao thức:

* Server Protocol Interpreter (Server-PI): chịu trách nhiệm quản lý kênh điều khiển trên server. Nó lắng nghe yêu cầu kết nối hướng tới từ users trên cổng dành riêng. Khi kết nối đã được thiết lập, nó sẽ nhận lệnh từ phía User-PI, trả lời lại, và quản lý tiến trình truyền dữ liệu trên server.
* Server DataTransfer Process (Server-DTP): làm nhiệm vụ gửi hoặc nhận file từ bộ phận User-DTP. Server-DTP vừa làm nhiệm thiết lập kết nối kênh dữ liệu và lắng nghe một kết nối kênh dữ liệu từ user. Nó tương tác với server file trên hệ thống cục bộ để đọc và chép file.

##### 2.2.4.3 Các tiến trình phía Client

* User Protocol Interpreter (User-PI): chịu trách nhiệm quản lý kênh điều khiển phía client. Nó khởi tạo phiên kết nối FTP bằng việc phát ra yêu cầu tới phía Server-PI. Khi kết nối đã được thiết lập, nó xử lý các lệnh nhận được trên giao diện người dùng, gửi chúng tới Server-PI, và nhận phản hồi trở lại. Nó cũng quản lý tiến trình User-DTP.
* User Data Transfer Process (User-DTP): là bộ phận DTP nằm ở phía người dùng, làm nhiệm vụ gửi hoặc nhận dữ liệu từ Server-DTP. User-DTP có thể thiết lập hoặc lắng nghe yêu cầu kết nối kênh dữ liệu trên server. Nó tương tác với thiết bị lưu trữ file phía client.
* User Interface: cung cấp giao diện xử lý cho người dùng. Nó cho phép sử dụng các lệnh đơn giản hướng người dùng, và cho phép người điều khiển phiên FTP theo dõi được các thông tin và kết quả xảy ra trong tiến trình.

#### 2.2.5 Trình tự truy cập và chứng thực FTP

Quy luật chứng thực trong FTP khá đơn giản, chỉ là cung cấp username/password. Trình tự của việc chứng thực như sau:

* Bước 1: Người dùng gửi một username từ User-PI tới Server-PI bằng lệnh USER. Sau đó password của người dùng được gửi đi bằng lệnh PASS.
* Bước 2: Server kiểm tra tên người dùng và password trong database người dùng của nó. Nếu người dùng hợp lệ, server sẽ gửi trả một thông báo tới người dùng rằng phiên kết nối đã được mở. Nếu người dùng không hợp lệ, server yêu cầu người dùng thực hiện lại việc chứng thực. Sau một số lần chứng thực sai nhất định, server sẽ ngắt kết nối.

Giả sử quá trình chứng thực đã thành công, server sau đó sẽ thiết lập kết nối để cho phép từng loại truy cập đối với người dùng được cấp quyền. Một số người dùng chỉ có thể truy cập vào một số file nhất định, hoặc vào một số loại file nhất định. Một số server có thể cấp quyền cho một số người dùng đọc và viết lên server, trong khi chỉ cho phép đọc đối với những người dùng khác. Người quản trị mạng có thể nhờ đó mà đáp ứng đúng các nhu cầu truy cập FTP.

Một khi kết nối đã được thiết lập, server có thể thực hiện các lựa chọn tài nguyên dựa vào nhận diện người dùng.

Ví dụ: trên một hệ thống nhiều người dùng, người quản trị có thể thiết lập FTP để khi có bất cứ người dùng nào kết nối tới, người dùng sẽ tự động được đưa tới “home directory” của chính người dùng.

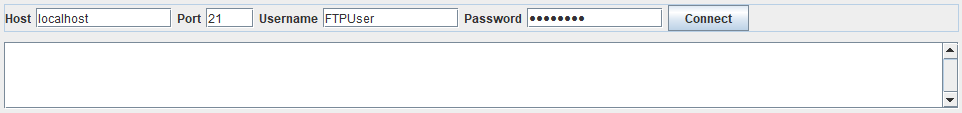
## CHƯƠNG 2. XÁY DỰNG CHƯƠNG TRÌNH VÀ DEMO

### 3.1 Các thành phần chính sử dụng trong chương trình

#### 3.1.1 Đăng nhập

Người dùng sẽ nhập các thông tin gồm username, password, host và port để kết nối đến FTP Server. Nếu thành công, khung bên dưới sẽ trả về giao diện cấu trúc thư mục trong Server để người dùng có thể tiến hành các tác vụ tiếp theo.

Nếu đăng nhập lỗi, sẽ xuất hiện thông báo lỗi và cho phép người dùng nhập lại.



Hình 18 : Đăng nhập vào chương trình

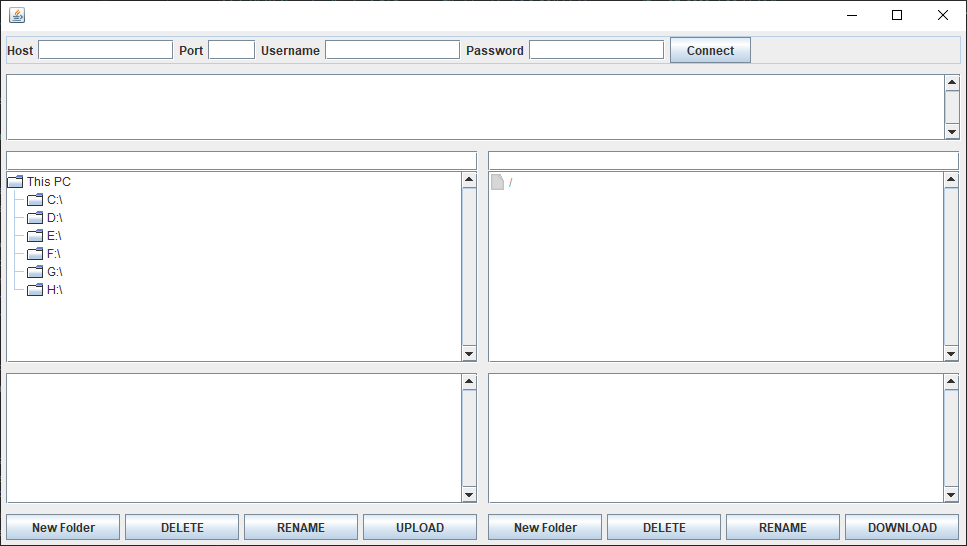
#### 3.1.2 Giao diện tổng thể

Giao diện dưới đây thể hiện tổng thể toàn bộ chương trình . Nhiệm vụ của toàn bộ chương trình bao gồm :

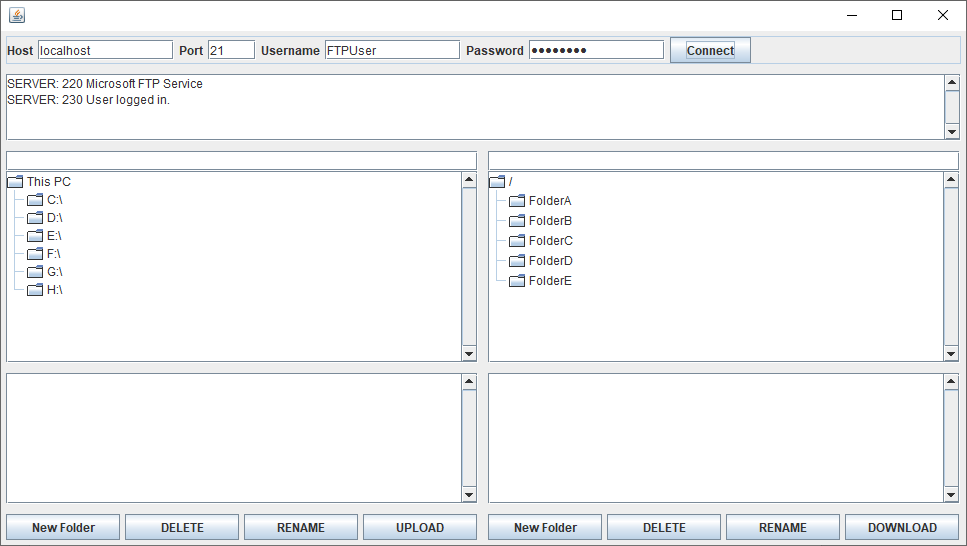
Nhập các thông tin vào các trường Username, Password, Host, Port. Nhấn vào nút Connect để tiến hành kết nối đến server.

* Sau khi đăng nhập, chương trình sẽ hiển thị cây thư mục ở phía server, người dùng có thể chọn đến thư mục để lưu trữ hoặc tải về các tệp tin từ cây thư mục phía bên server ở bên phải giao diện.
* Thư mục hiện tại mà người dùng đang chọn sẽ được hiển thị ở phía trên mỗi cây thư mục tương ứng.
* Nút “New Folder” : Cho phép người dùng tạo thêm một thư mục mới ở phía client hoặc bên phía server.
* Nút “DELETE”: Cho phép người dùng xóa một thư mục hoặc file ở phía client hoặc server.
* Nút “RENAME”: Cho phép người dùng đổi tên thư mục hoặc file ở bên client hoặc server.
* Nút UPLOAD: Cho phép người dùng upload file hoặc thư mục ở list bên dưới đến server;
* Nút “DOWNLOAD”: Cho phép người dùng downlaod một file hoặc thư mục từ server về client.

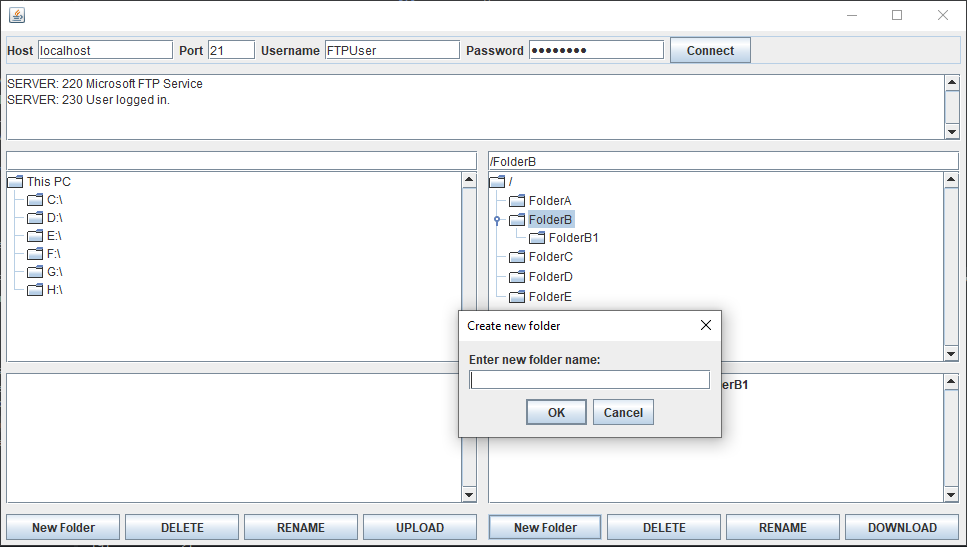
### 3.2 Chạy demo chương trình



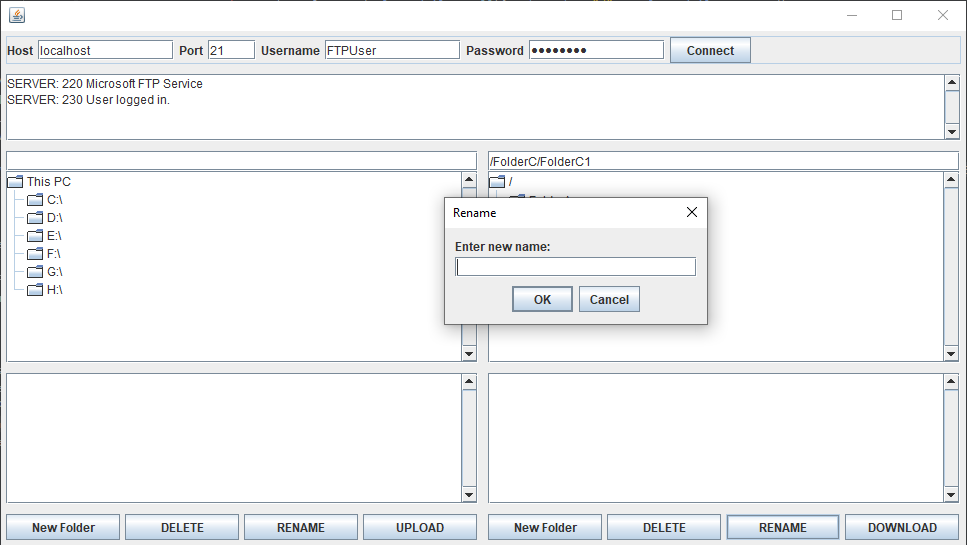
Hình 19 : Giao diện khi chưa đăng nhập



Hình 7 : Giao diện sau khi đăng nhập



Hình 8 : Thêm mới một thư mục tại vị trí chỉ định



Hình 9 : Chỉnh sửa tên của thư mục / tệp tin

## CHƯƠNG 3. KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

### 3.1 Kết quả đạt được

* Ứng dụng có giao diện trực quan, rõ rang. Tương đối dễ để sử dụng.
* Hệ thống có quản lý trực quan các thư mục từ phía Server, giúp dễ quản lý.
* Tích hợp ngôn ngữ lập trình Java , có khả năng tuỳ biến và nâng cao hiệu suất sử dụng.

### 3.2 Vấn đề chưa giải quyết được

* Hệ thống vẫn chưa có khả năng quản lý thư mục trực quan từ phía máy tính người dùng
* Chưa áp dụng thành công bảo mật thông tin khi sử dụng giao thức FTP.
* Chưa có chức năng upload/download nhiều file cùng lúc mà chỉ cho phép chọn một.

### 3.3 Hướng phát triển

* Cho phép tải nhiều dữ liệu được chọn cùng một lúc.
* Tích hợp thêm khả năng bảo mật thông tin.
* Cho phép quản lý thư mục của máy tính người dùng ngay trên chương trình ứng dụng.

TÀI LIỆU THAM KHẢO

[1] Slide bài giảng Lập trình mạng của thầy Mai Văn Hà.

[2] <https://afteracademy.com/blog/what-is-a-tcp-3-way-handshake-process>

[3] <https://viblo.asia/p/tim-hieu-ve-http-hypertext-transfer-protocol-bJzKmgewl9N>

[4] Giáo trình hệ ñiều hành, chương 6: Deadlock

[5] <https://vi.wikipedia.org/wiki/Deadlock>

[6] <https://en.wikipedia.org/wiki/Network_socket>

[7] <https://vi.wikipedia.org/wiki/TCP/IP>