**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

──────── \* ───────



**BÁO CÁO BÀI TẬP LỚN**

**MÔN LẬP TRÌNH MẠNG**

**Đề tài :** Xây dựng chương trình mô phỏng vòng loại cuộc thi Binary Brain

Giảng viên hướng dẫn : Thầy **Bùi Trọng Tùng**

Sinh viên thực hiện : **Lê Đình Xuân Bách- AS2**

**Nguyễn Văn Thọ- AS2**

*Hà Nội, tháng 11 năm 2010*

# MỤC LỤC

[LỜI MỞ ĐẦU 3](#_Toc278525390)

[**1)** **Giới thiệu về đề tài (mục tiêu, hướng giải quyết,..)** 3](#_Toc278525391)

[**2)** **Phân công công việc:** 4](#_Toc278525392)

[**3)** **Tóm tắt các chương:** gồm 3 chương. 4](#_Toc278525393)

[CHI TIẾT CÁC CHƯƠNG 5](#_Toc278525394)

[**I.** **Cơ sở lý thuyết** 5](#_Toc278525395)

[1) Socket là gì ? 5](#_Toc278525396)

[2) Các loại Sockets chính trong TCP/IP 5](#_Toc278525397)

[**II.** **Xây dựng chương trình** 8](#_Toc278525398)

[1) Cấu trúc dữ liệu và thư viện hỗ trợ: 8](#_Toc278525399)

[2) Giao tiếp giữa client và server trong chức năng 9](#_Toc278525400)

[**III.** **Cài đặt, kiểm thử** 16](#_Toc278525401)

[1. Giao diện mở đầu: 16](#_Toc278525402)

[2. Login, register: 17](#_Toc278525403)

[3. Exception: 17](#_Toc278525404)

[4. Hướng phát triển: 18](#_Toc278525405)

[**IV.** **Kết luận, tài liệu tham khảo** 19](#_Toc278525406)

[1) Kết luận 19](#_Toc278525407)

[2) Tài liệu tham khảo 19](#_Toc278525408)

# LỜI MỞ ĐẦU

Ngày nay máy tính đã trở thành một phần không thể thiếu trong cuộc sống của mỗi chúng ta. Với sự phát triển nhanh chóng như vậy, một máy tính không chỉ đơn thuần là giải quyết bài toán ở một nơi cố định, độc lập mà chúng còn phải giao tiếp với nhau để làm việc hiệu quả hơn. Với tư cách là người tạo ra ngôn ngữ để máy tính giao tiếp,nên với một kỹ sư CNTT thì những kiến thức về lập trình mạng ngày càng trở nên quan trọng.

Nhờ sự giúp đỡ, giảng dạy nhiệt tình của thầy Bùi Trọng Tùng, chúng em đã có những kiến thức cơ bản về Lập trình mạng. Nhằm trau dồi những kiến thức đó và làm cho nó trở nên thực tế hơn, nhóm em đã quyết định làm đề tài “Xây dựng chương trình mô phỏng vòng loại cuộc thi Binary Brain”.

Tuy vậy do trình độ hiểu biết còn có hạn nên bài tìm hiểu của nhóm không thể không có những sai sót. Em rất mong nhận được những lời nhận xét quý báu của thầy để bài của nhóm em được hoàn thiện hơn.

Em xin chân thành cảm ơn.

1. **Giới thiệu về đề tài (mục tiêu, hướng giải quyết,..)**
2. Tên đề tài: Xây dựng chương trình mô phỏng vòng sơ loại cuộc thi Binary Brain
3. Thực tế: Vòng sơ loại Binary Brain có 2 phần thi:

01. Thi kiến thức chung về CNTT:  
  Hình thức: Trắc nghiệm. Các đội trả lời bằng hình thức giơ bảng.  
  Nội dung: Các câu hỏi được đặt ra xoay quanh các ứng dụng phong phú của CNTT trong cuộc sống xã hội. Có 10 câu hỏi được đưa ra, sau 15 giây suy nghĩ, các đội đồng loạt đưa ra câu trả lời của mình bằng hình thức giơ bảng. 10 điểm cho một câu trả lời đúng.

02. Thi kiến thức chuyên môn CNTT  
  Hình thức: Trắc nghiệm. Các đội bấm chuông giành quyền trả lời.  
   Nội dung: Các câu hỏi được đặt ra xoay quanh các kiến thức chuyên ngành CNTT. Có 05 câu hỏi được đặt ra. Trong 15 giây suy nghĩ mà vẫn chưa có đội nào bấm chuông trả lời, quyền trả lời sẽ thuộc về khán giả. 20 điểm cho một câu trả lời đúng. Bấm chuông trả lời sai bị trừ đi 05 điểm.

1. Áp dụng vào chương trình:

Xây dựng chương trình gồm 2 phần :

+ Phần 1: Cả 4 client đều được trả lời, mỗi câu trả lời đúng được 10 điểm, trả lời sai không bị trừ điểm.

+ Phần 2: Cả 4 client đều được trả lời, mỗi câu trả lời đúng được 20 điểm, trả lời sai bị trừ 5 điểm.

1. **Phân công công việc:**

Nhóm gồm 2 thành viên:

Lê Đình Xuân Bách

Nguyễn Văn Thọ

Các công việc chính mà thành viên trong nhóm đã phân công nhau làm như sau:

Công việc của Thọ:

* Tìm hiểu yêu cầu bài toán
* Đưa ra các chức năng mà chương trình cần có
* Viết chương trình phía client
* Xử lí file đầu vào
* Kiểm thử chương trình
* Báo cáo

Công việc của Bách:

* Xây dựng nguyên mẫu hàm, thiết kế chương trình
* Viết các hàm phía server, xây dựng kết nối client-server
* Kiểm thử chương trình, xây dựng chương trình theo hướng phát triển (giao diện GTK)
* Báo cáo

1. **Tóm tắt các chương:** gồm 3 chương.

Chương 1: Cơ sở lí thuyết

Chương 2: Xây dựng chương trình

Chương 3: Cài đặt và kiểm thử

# CHI TIẾT CÁC CHƯƠNG

1. **Cơ sở lý thuyết**
2. **Socket là gì** ?

- Ta biết rằng mỗi máy tính muốn giao tiếp với nhau cần phải có một địa chỉ duy nhất mà hiện nay thường được dùng là điạ chỉ IP.

- Với địa chỉ IP, chúng ta có thể xác định ra một máy tính duy nhất trên mạng, tuy nhiên khi một máy tính chạy nhiều chuơng trình, thì các chương trình đó được phân biệt với nhau bởi khái niệm port.

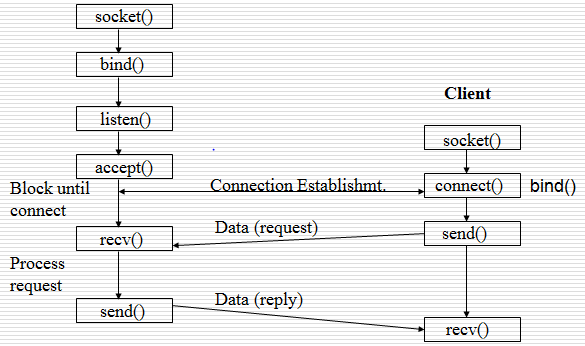
- Như vậy kết hợp giữa IP và port thì chúng ta mới xác định được một dịch vụ cụ thể để truy nhập đến máy chủ🡪 Socket.

- Một Socket là một trừu tượng mà thông qua nó một ứng dụng có thể gửi và nhận dữ liệu.

- Địa chỉ Socket bao gồm cả tên máy chủ và số hiệu cổng.

1. **Các loại Sockets chính trong TCP/IP**
2. **Stream sockets: sử dụng phương thức TCP (truyền thông tin cậy, có bắt tay ba bước) để tăng độ tin cậy.**

- Giao thức TCP tạo kết nối giữa clients và server. Nó cũng làm tăng độ tin cậy bằng cách: Khi TCP gửi dữ liệu, nó yêu cầu phải có ACK trả về. Cả gửi dữ liệu và nhận dữ liệu đều trên một kết nối (full-duplex).



Server

*+socket*: khởi tạo socket(buy a phone)

*+bind*: gán địa chỉ cho socket (get a telephone number)

*+listen*: lắng nghe một kết nối (call waiting allowance)

*+accept*: server đồng ý yêu cầu kết nối từ một máy client(answer phone)

Có 3 cơ chế cơ bản để accept, từ đó phân loại ra 3 loại server:

*-* Iterating server: ở một thời điểm chỉ có duy nhất 1 socket được mở.

- Forking server: sau khi server chấp nhận 1 kết nối, nó chuyển accept cho tiến trình con. Loại server này dễ bị tấn công Dos khi có người cố tình gửi quá nhiều yêu cầu kết nối đến.

- Concurrent single server: server tạo ra 1 hàng đợi, các client đưa hết vào hàng đợi, chưa accept vội, nếu client sẵn sàng gửi dữ liệu thì gọi ra một tiến trình để trao đổi dữ liệu với client.

*+connect*: client yêu cầu một kết nối đến server (call)

*+send*: gửi dữ liệu (speak)

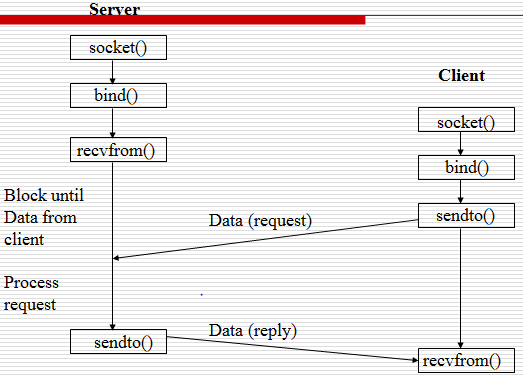
*+recv* : nhận dữ liệu (listen)

*+close*: đóng socket (end the call)

1. **Datagram sockets: sử dụng phương thức UDP (không tin cậy), tập trung cố gắng truyền được càng nhiều dữ liệu càng tốt.**

- Gửi những dữ liệu ngắn được gọi là [*datagram*](http://vi.wikipedia.org/w/index.php?title=Datagram&action=edit&redlink=1) tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận như [TCP](http://vi.wikipedia.org/wiki/TCP) làm.

- Các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.



Ở đây không có quá trình bắt tay 3 bước.

1. **Xây dựng chương trình**
2. **Cấu trúc dữ liệu và thư viện hỗ trợ:**

a) Trong file supportLib.h

- Có một cấu trúc để lưu thông tin username và password để dùng khi đăng nhập vào hệ thống.

*typedef struct np{*

*char name[20]; //username để đăng nhập*

*char pass[20]; //password để đăng nhập*

*}NamePass;*

- Cấu trúc để lưu thông tin về người chơi

*typedef struct us{*

*int FD; //socket khi user đăng nhập thành công.*

*char name[20]; //tên đội chơi*

*int mark; //số điểm của đội*

*int lost;//cờ báo user bị out ra lúc đang chơi*

*}User;*

- Cấu trúc để lưu thông tin về câu hỏi :

*typedef struct qa{*

*char question[1000]; // nội dung câu hỏi*

*char ans[5]; //đáp án đúng*

*char p[5]; //phần thi: phần 1 hay 2*

*int toAns;/cờ báo time out cho câu hỏi này*

*}QA;*

*-*Cấu trúc file lưu tên và password đội chơi:

Gồm nhiều dòng,mỗi dòng là:

*Username+password*

-Cấu trúc file lưu question QA.txt:

Gồm 3 phần :

+ chỉ ra phần thi 1 hay 2

+ câu hỏi

+ đáp án đúng

Ví dụ:

*P1\_ // đây là phần thi 1*

*Q\_ //bắt đầu phần câu hỏi*

*Hiệu trưởng đầu tiên của ĐHBKHN là ai?*

*Tra loi*

*A: Tạ Quang Bửu*

*B: Nguyễn Trọng Giảng*

*A\_ //bắt đầu phần đáp án đúng*

*A*

b) Trong file sockhelp.h

-Hàm ***makeconnection()*** tạo kết nối, phía client

-Hàm ***sock\_gets()*** : nhận dữ liệu dạng chuỗi.

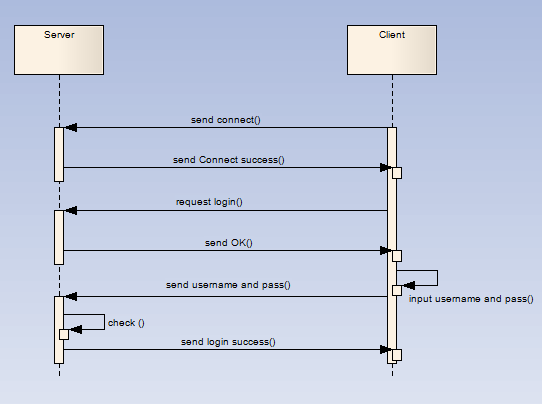
-Hàm ***sock\_puts()***: gửi dữ liệu dạng chuỗi, kết thúc khi gặp kí tự ‘\n’

1. **Giao tiếp giữa client và server trong chức năng**

**Chương trình sử dụng hoàn toàn giao thức TCP và dùng select server.**

1. **Login:**

Biểu đồ tuần tự:



Hoạt động như sau:

**Client:**

* Hiển thị menu chọn login hay register
* Sau khi người dùng chọn, cho phép người dùng nhập username và password
* Checkvalid username và password (không có kí tự trắng trong username,độ dài <20….)
  + Nếu không valid cho phép nhập lại
  + Nếu valid gửi dữ liệu lên server dưới dạng chuỗi như sau:

Username+password+type

(Type là lựa chọn login hay hay register)

* Gửi dữ liệu chuỗi này lên server
* Chờ nhận thông báo trả về từ server: BAD hoặc OK
  + Nếu là BAD thì cho phép thực hiện lại lựa chọn.
  + Nếu là OK chờ nhận câu hỏi từ server.

**Server:**

* Nhận dữ liệu từ client dưới dạng chuỗi
* Xử lí dữ liệu nhận: có 2 kiểu dữ liệu mà server nhận được từ client đó là Request hay Answer.

Hàm xử lí dữ liệu nhận là ***read\_socks().***

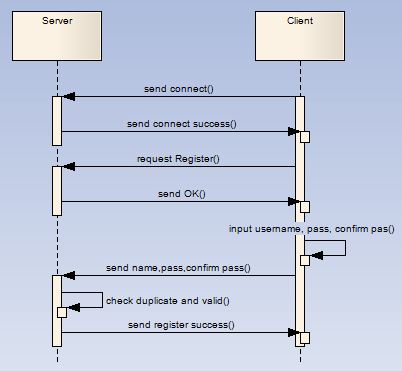
Hàm ***checkRequestAns()***sẽ xét xem dữ liệu nhận được từ client là Request hay Answer.

* + Nếu là Request: xử lí đăng nhập hoặc đăng kí
    - Dữ liệu server nhận được có dạng chuỗi:

Username+password+type

* + - Để xác định xem Request là đăng nhập hay đăng kí ta dùng hàm ***cutNamePass()*** cắt dữ liệu chuỗi server nhận được từ client, biến type xác định được Request là login hay register.
    - Nếu đăng nhập thành công, lưu lại thông tin user đăng nhập vào mảng ***listUs*** có kiểu ***User*** với các thông số tương ứng, trong đó đặc biệt lưu ý thông số ***lost*** cho phần xử lí **ngoại lệ** (**exception** ) sau này.
  + Nếu là Answer :
    - Dữ liệu nhận được là chuỗi chứa câu trả lời mà client đưa ra.
    - xử lí tính điểm trả lời câu hỏi bằng hàm ***calculateMark()***

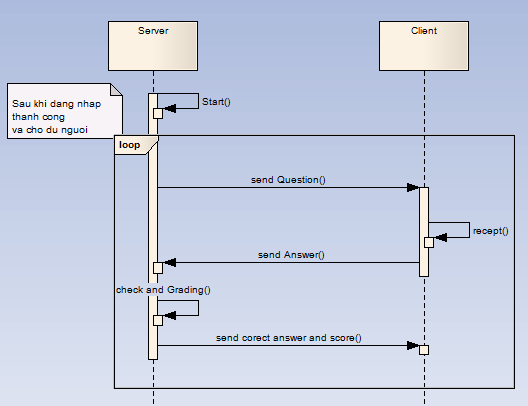
1. **Register**



Hoạt động register tương tự login.

1. **Tiến hành cuộc thi**

Khi các client đã đăng nhập thành công. Server sẽ chờ cho đến khi đủ 4 đội chơi tham gia thì sẽ cho bắt đầu cuộc thi. Khi đó các client có vai trò như nhau.



Hoạt động như sau:

Sau khi đăng nhập hệ thống thành công, đã sẵn sàng cho cuộc thi:

**Client:**

* Chờ nhận câu hỏi từ server.
* Gửi câu trả lời cho server
* Xử lí nếu sau 15 giây mà không gửi câu trả lời cho server thì tự động thoát hàm nhập câu trả lời ***scanf()*** bằng cách dùng hàm ***bkhit()*** trong đó dùng hàm select để lắng nghe ở cổng stdin (cổng 0) trong vòng timeout là 15s, trả về kết quả là 1 nếu cổng stdin là ready,trả về là 0 nếu cổng stdin không ready trong vòng 15s.
* Nhận tín hiệu kết thúc từ server để dừng kết nối

**Server:**

* Đọc toàn bộ câu hỏi từ file vào mảng ***listQA*** có cấu trúc ***QA.***
* Gửi câu hỏi lần lượt trong mảng ***listQA***.
* Nhận câu trả lời từ client
* Xử lí tính điểm qua hàm ***calculateMark()***
* Sau 15 giây, server tự động chuyển sang câu hỏi mới và đặt ***toAns*** cờ báo cho câu hỏi trước đã timeout.
* Gửi tín hiệu kết thúc đến tất cả các client sau khi kết thúc toàn bộ các câu hỏi và dừng kết nối, kết thúc cuộc chơi.

**Xử lí ngoại lệ (Exception):** Đây là một phần khá hay mà nhóm đã nghiên cứu kĩ để xử lí phần này một cách hiệu quả nhất.

Ngoại lệ có thể xảy ra như sau:

* Giai đoạn login, sau khi đã login thành công, một hoặc 1 vài client bị ngắt kết nối. **(exception 1)**
* Giai đoạn gửi câu hỏi, trước hoặc sau lúc gửi câu hỏi đến client thì một client hoặc 1 vài client bị ngắt kết nối **(exception 2)**. Lúc này tắt cả client đã đăng nhập đầy đủ vào hệ thống và đang chờ cầu hỏi được gửi từ server.

Khi xảy ra ngoại lệ, để chương trình vẫn có thể tiếp tục, điểm của các đội chơi không bị thay đổi ta cần đưa ra phương pháp xử lí ngoại lệ thích hợp.

**Xử lí exception 1:**

* Khi xảy ra ngoại lệ này sẽ thông báo connection lost ở FD bao nhiêu, động thời thực hiện ở user có số hiệu FD tương ứng hành động sau:
  + Đưa cổng này ra khỏi tập các cổng đang lắng nghe.
  + Bật cờ hiệu ***lost***  về giá trị 0 báo hiệu user này bị connection lost
* Cho phép đội chơi bị out đăng nhập lại bằng hàm ***acceptForClient()*** trong đó check xem tất cả các user đã đăng nhập đầy đủ chưa bằng hàm :
  + ***checkConnectionLost\_connectList()***
  + ***checkConnectionLost()***: check trong ***listUser***

xem cờ hiệu ***lost*** trong mỗi user đã được đưa về 1 hay chưa (lost=1 chứng tỏ đã login thành công,hiện đang sử dung hệ thống, lost=0 chứng tỏ chưa login và đang không sử dụng hệ thống)

* + Nếu đã đăng nhập đầy đủ 4 client vào hệ thống thì cho phép hệ thống tiếp tục bắt đầu chạy phần gửi câu hỏi.

**Xử lí exception 2:**

* Lúc xảy ra ngoại lệ này, chương trình đã bắt đầu tính điểm cho các đội chơi, vấn đề đặt ra là làm sao cho hệ thống chờ user bị out đăng nhập lại, số điểm của tất cả các đội trước lúc xảy ra exception sẽ không bị thay đổi (quan trọng là ở user bị out )
* Khi xảy ra ngoại lệ này sẽ thông báo connection lost ở FD bao nhiêu, động thời thực hiện ở user có số hiệu FD tương ứng hành động sau:
  + Đưa cổng này ra khỏi tập các cổng đang lắng nghe.
  + Bật cờ hiệu ***lost***  về giá trị 0 báo hiệu user này bị connection lost
* Tạo một luồng (**thread**) con gọi đến hàm ***lostThread()***

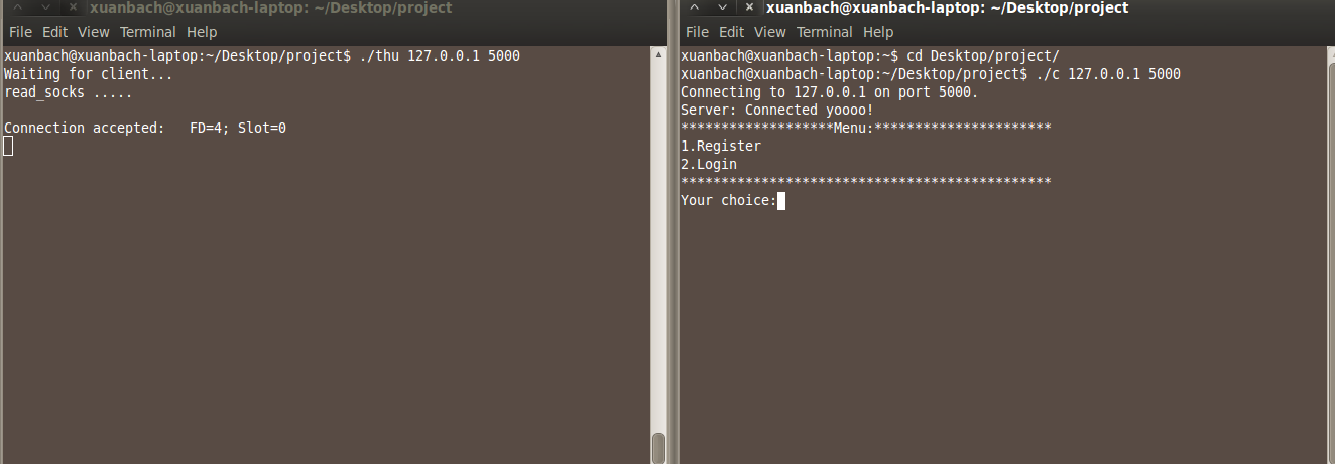
Trong đó hàm ***lostThread()*** chạy vòng lặp vô hạn gọi đến hàm ***acceptForClient().*** Nghĩa là luồng con này luôn luôn thực hiện:

* + Lắng nghe xem cổng nào bị connection lost
  + Cho phép user đăng nhập lại với cổng đăng nhập cũ,tên đăng nhập cũ. Nếu user đang thực hiện đăng nhập lại có FD trùng với FD của một user trong ***listUser*** thì ta không tạo mới 1 user mà chỉ bật lại cờ hiệu ***lost*** cho user tương ứng đó là 1 (nghĩa là đã login trở lại thành công). Lúc đó số điểm của user trước lúc xảy ra exception đến lúc đăng nhập lại sẽ vẫn được giữ nguyên.
  + Lắng nghe tất cả các cổng đang connect đến server hiện tại , để nhận câu trả lời và xử lí tính điểm.
* Trong lúc **luồng con** thực hiện, **luồng chính** vẫn chạy và thực hiện:
  + Chờ lệnh gửi câu hỏi đến các client bằng dữ liệu nhập từ bàn phím (send question? y or n)
  + Nếu nhập vào “y” thì thực hiện gửi câu hỏi đến các client.
  + **Lưu ý**: trong lúc này luồng con đang thực hiện lắng nghe các client để nhận câu trả lời.

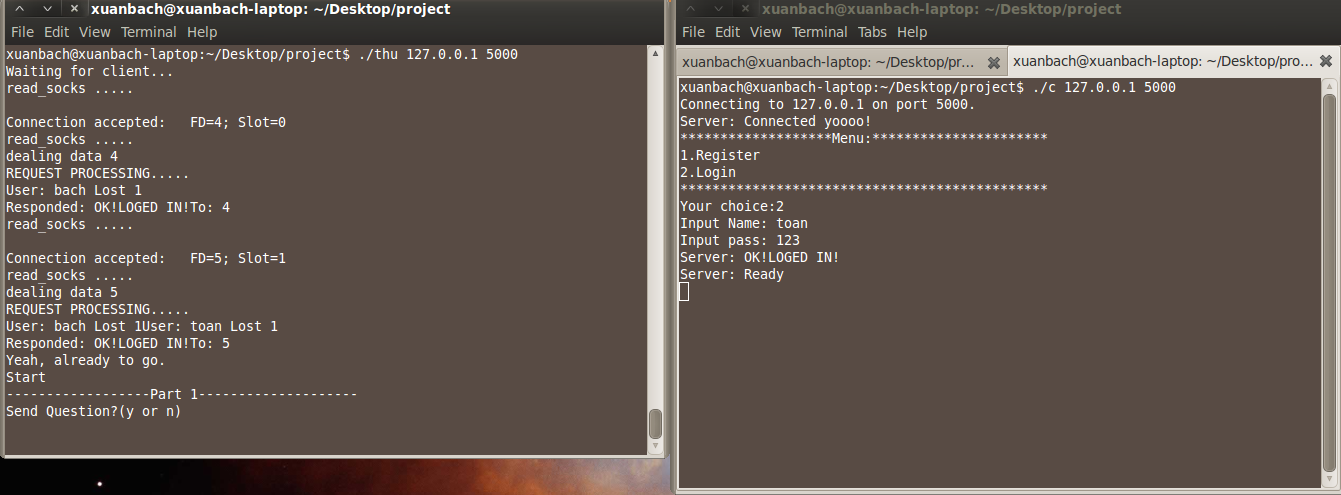
***Trên đây là sợ bộ cách giải quyết bài toán mà nhóm đã thực hiện.***

***Sau đây là phần cài đặt kiểm thử,giao diện chương trình.***

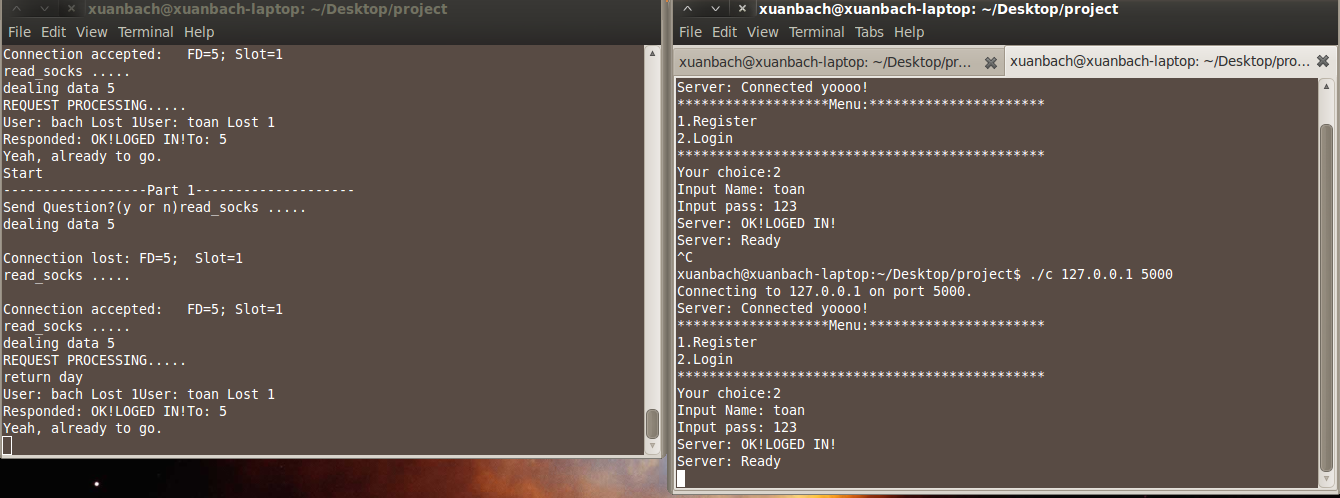
1. **Cài đặt, kiểm thử**
2. **Giao diện mở đầu:**



1. **Login, register:**



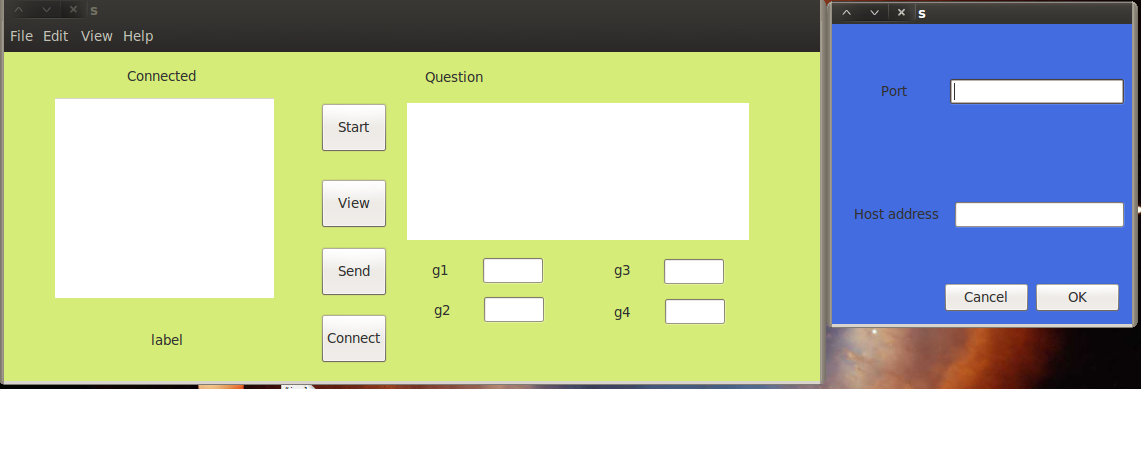
1. **Exception:**



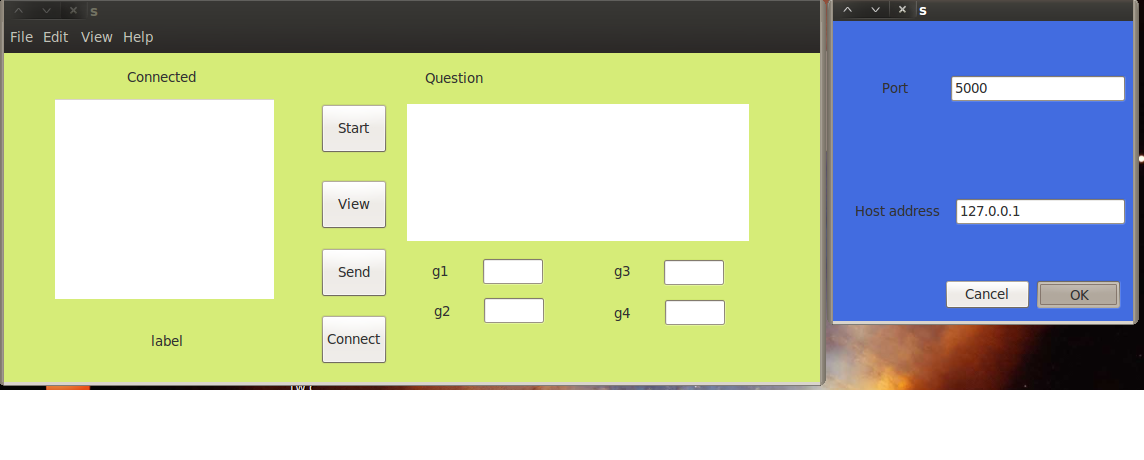
### Hướng phát triển:

Nhóm đưa ra hướng phát triển xây dựng giao diện người dùng với GTK như sau:

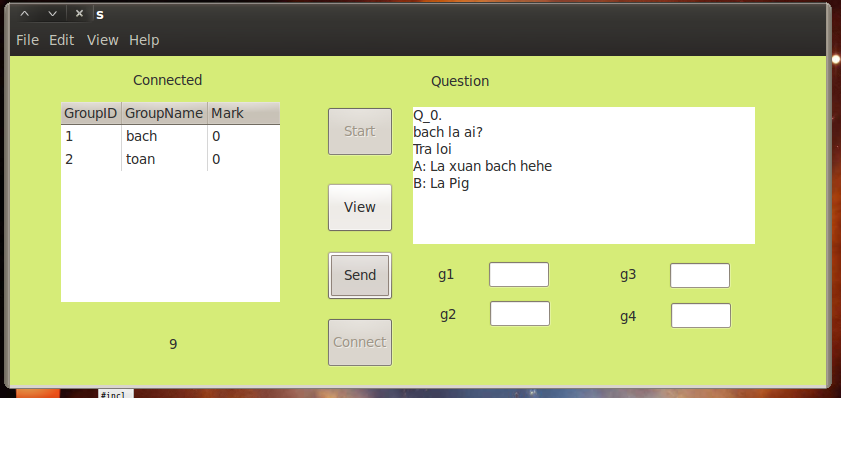
1. Giao diện mở đầu:



1. Mở port, lắng nghe client:



1. Gửi câu hỏi, xử lí tính điểm, đồng hồ báo timeout (label sẽ thay đổi từ 15->0 như là đồng hồ chạy timeout )sau khi đã hoàn thành khâu đăng nhập.



1. **Kết luận, tài liệu tham khảo**
2. **Kết luận**

Qua việc tiến hành làm bài tập lớn, nhóm đã thu được nhưng kết quả sau:

* Hiểu về quy cách lập trình socket
* Đặc biệt: sử dụng select server, nhóm đã hiểu hơn về đặc tính của loại server này.
* Hướng phát triển của nhóm là xây dựng chương trình binary brain có khả năng ứng dụng vào thực tiễn cho cuộc thi binary brain

1. **Tài liệu tham khảo**

[1] Sile bài giảng của thầy Bùi Trọng Tùng.

[2] Websites về lập trình Socket:

<http://www.csc.villanova.edu/~mdamian/Sockets/TcpSockets.htm#client-server>

<http://shoe.bocks.com/net/>

[3] Sách “ *TCP/IP Sockets in C* ” của Michael J. Donahoo và Kenneth L. Canvert