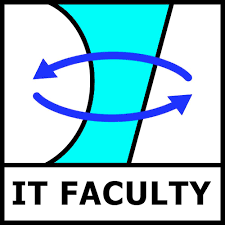


**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------



***BÁO CÁO ĐỒ ÁN*  
CƠ SỞ NGÀNH MẠNG**

***ĐỀ TÀI***

1. **Giao tiếp các tiến trình bằng đường ống PIPE**
2. **Tìm hiểu và sử dụng kỹ thuật lập trình Socket**

**xây dựng chương trình xem điểm thi theo mô hình Client - Server.**



**GVHD**: **ThS.** **NGUYỄN VĂN NGUYÊN**

**SVTH** : **Lê Thị Hà Bình**

**MSSV** : 102140113

***Đà Nẵng****, 2018*

MỞ ĐẦU

Đồ án cơ sở ngành mạng là một trong những đồ án quan trọng của ngành Công Nghệ Thông Tin. Thông qua đồ án này, chúng em – là những sinh viên của trường đại học Bách Khoa Đà Nẵng sẽ đạt được những kiến thức nền tảng và có cơ hội được nghiên cứu chuyên sâu hơn về bộ môn *Lập trình mạng và Nguyên lí hệ điều hành.*

Trong đồ án này, về phần *Nguyên lí hệ điều hành*, em nghiên cứu và tìm hiểu về cách thức giao tiếp giữa các tiến trình bằng cơ chế đường ống PIPE, còn phần *Lập trình mạng*, đề tài được chọn là xây dựng chương trình xem điểm thi theo mô hình Client - Server. Với sự đồng hành và hỗ trợ nhiệt tình của các thầy cô bộ môn ngành Mạng cùng sự nỗ lực cố gắng của bản thân, bản báo cáo *Đồ án cơ sở ngành mạng* đã hoàn thành đúng với thời hạn được giao.

Đặc biệt em xin trân trọng cảm ơn thầy Nguyễn Văn Nguyên đã hướng dẫn và chỉ bảo tận tình trong quá trình nghiên cứu đồ án này.

***Sinh viên thực hiện***

Lê Thị Hà Bình

MỤC LỤC

[**MỞ ĐẦU** 1](#_Toc501995693)

[**PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH** 3](#_Toc501995694)

[CHƯƠNG 1. CỞ SỞ LÍ THUYẾT 4](#_Toc501995695)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ 5](#_Toc501995696)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ](#_Toc501995697) 9

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 11](#_Toc501995698)1

[**PHẦN II: LẬP TRÌNH MẠNG** 12](#_Toc501995699)2

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 12](#_Toc501995700)2

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 15](#_Toc501995701)5

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 17](#_Toc501995702)7

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 17](#_Toc501995703)9

[**PHỤ LỤC** 20](#_Toc501995704)

[**KẾT LUẬN CHUNG** 3](#_Toc501995705)1

# PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

**TIÊU ĐỀ: Giao tiếp giữa các tiến trình bằng cơ chế đường ống**

Yêu cầu đề tài:

1. Giới thiệu tiến trình trong Linux, các hoạt động của tiến trình, cấu trúc tiến trình, giao tiếp giữa các tiến trình, các cơ chế giao tiếp và giới thiệu cơ chế liên lạc bằng đường ống Pipe.

2. Tạo đường ống giao tiếp ( có thể sử dụng hàm Fork()), các loại pipe

3. Viết chương trình gồm 2 quá trình. Quá trình thứ nhất cho người dùng nhập vào từ bàn phím một chuỗi biểu diễn các phép tính gồm các phần tử +,-,(,). độ ưu tiên của các phép tính trong ngoặc(cặp dấu (&)) là cao nhất, phép +,và – có cùng độ ưu tiên. Ví dụ:

1+2+(2-3-4)-((3+4)-5)

(1+(-2)-((3+4)-5))

Sau đó truyền chuỗi dữ liệu này sang quá trình thứ hai. Quá trình thứ hai thực hiện tính toán và trả về cho quá trình thứ nhất để thể hiện cho người sử dụng biết

5. Viết báo cáo theo mẫu Bộ môn mạng và Truyền thông

6. Gặp giáo viên hướng dẫn, làm đúng yêu cầu của giáo viên hướng dẫn và báo cáo tiến độ đúng thời gian

7. Ngôn ngữ dùng để viết chương trình C trên Linux hoặc ubuntu

8. Làm 10 đến 15 slide để bảo vệ sau khi được giáo viên hướng dẫn cho phép.

## **CHƯƠNG 1. CỞ SỞ LÍ THUYẾT**

**1.1. TIẾN TRÌNH TRONG LINUX**

* Tiến trình là một chương trình đang xử lý, sở 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
* Cấu trúc tiến trình:  
  Trong quá trình thực hiện,một tiến trình được đặc trưng bởi nhiều thuộc tính do hệ thống duy trì như:
* Trạng thái của nó.
* Định danh của nó.
* Các giá trị của các thanh ghi, bao gồm cả bộ đếm chương trình.
* Mã định danh người sử dụng có tên mà tiến trình đang thực hiện.
* Thông tin được kernel sử dụng để thiết lập lịch biểu của các tiến trình (thứ tự ưu tiên, v.v…).
* Các thao tác điều khiển tiến trình:
* Tạo lập tiến trình bằng hàm system()
* Thay thế tiến trình hiện hành với hàm exec()
* Nhân bản tiến trình với hàm fork()
* Kiểm soát và đợi tiến trình con
* Đón xử lý tín hiệu khi tiến trình con kết thúc
* Bỏ rơi tiến trình con

**1.2. CƠ CHẾ GIAO TIẾP CÁC TIẾN TRÌNH**

Một tiến trình là một thực thể hoạt động của hệ điều hành thực thi các chương trình. Trong thực tế, có những tiến trình phức tạp và cần phải nhiều tiến trình giao tiếp để hoàn thành một công việc. Để làm việc cùng nhau, các tiến trình cần phải trao đổi dữ liệu. Vì vậy, cần phải có nhiều cơ chế giao tiếp interprocess (IPC) khác nhau. Một trong những cơ chế IPC cơ bản nhất là đường ống, tượng trưng cho dữ liệu tuần tự giữa các quá trình trong một đường ống.

**1.3. CƠ CHẾ GIAO TIẾP BẰNG ĐƯỜNG ỐNG – PIPE**

Các tiến trình chạy độc lập có thể chia sẻ hoặc chuyển dữ liệu cho nhau xử lý thông qua cơ chế đường ống (pipe).

* Một pipe là một thiêt bị truyền thông tuần tự, dữ liệu có thể đọc (read) từ pipe cùng lúc ghi (write) lên pipe
* Pipe còn được dùng để liên lạc giữa hai thread trong một tiến trình hay giữa tiến trình cha và tiến trình con
* Trên đường ống dữ liệu chỉ có thể chuyển đi theo một chiều, dữ liệu vào đường ống tương đương với thao tác ghi (pipe write), lấy dữ liệu từ đường ống tương đương với thao tác đọc (pipe read). Dữ liệu được chuyển theo luồng (stream) theo cơ chế FIFO.
* Một pipe là một kênh liên lạc trực tiếp giữa hai tiến trình: dữ liệu xuất của tiến trình này được chuyển đến làm dữ liệu nhập cho tiến trình kia dưới dạng một dòng các byte.
* Khi một pipe được thiết lập giữa hai tiến trình, một trong chúng sẽ ghi dữ liệu vào pipe và tiến trình kia sẽ đọc dữ liệu từ pipe. Thứ tự dữ liệu truyền qua pipe được bảo toàn theo nguyên tắc FIFO.

**1.4. ĐƯỜNG ỐNG GIAO TRIẾP TRAO ĐỔI DỮ LIỆU HAI CHIỀU**

Sử dụng cơ chế giao tiếp đường ống hai chiều dễ dàng cho cả hai phía tiến trình cha và tiến trình con. Trong cơ chế giao tiếp hai chiều bằng đường ống thì thường người ta sẽ dùng hai đường ống một chiều vì sẽ dễ xử lý hơn là sử dụng một đường ống. Khi sử dụng một đường ống phải tính toán sao cho tránh được tình huống nghẽn do cả hai phía cùng ghi dữ liệu vào đường ống. Trong đồ án này, nhóm chúng em sử dụng hai đường ống một chiều để thực hiện giao tiếp hai chiều giữa tiến trình cha và tiến trình con.

## **CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ**

Bài toán giao tiếp hai chiều giữa tiến trình cha và tiến trình con là một bài toán rất hay gặp và được ứng dụng rất nhiều. Tuy nhiên đối với dạng bài toán này nếu chỉ dùng một đường ống thì phải xử lý phức tạp, nguyên nhân là vì cơ chế đóng mở đầu khi đọc ghi của pipe. Chính vì vậy một giải pháp đơn giản cho bài toán này đó là sử dụng hai đường ống một chiều để giao tiếp.

Trong đó, một đường ống chỉ được dùng chuyên biệt cho việc nhận dữ liệu từ tiến trình cha, và đường ống còn lại được dùng để ghi ngược dữ liệu từ tiến trình con đến tiến trình cha.

**Bài toán như sau:**

Viết chương trình gồm 2 quá trình. Quá trình thứ nhất cho người dùng nhập vào từ bàn phím một chuỗi biểu diễn các phép tính gồm các phần tử +,-,(,). độ ưu tiên của các phép tính trong ngoặc(cặp dấu (&)) là cao nhất, phép +,và – có cùng độ ưu tiên. Ví dụ:

1+2+(2-3-4)-((3+4)-5)

(1+(-2)-((3+4)-5))

Sau đó truyền chuỗi dữ liệu này sang quá trình thứ hai. Quá trình thứ hai thực hiện tính toán và trả về cho quá trình thứ nhất để thể hiện cho người sử dụng biết

**Tóm tắt bài toán:**

**ĐẦU VÀO:** Chuỗi phép tính str

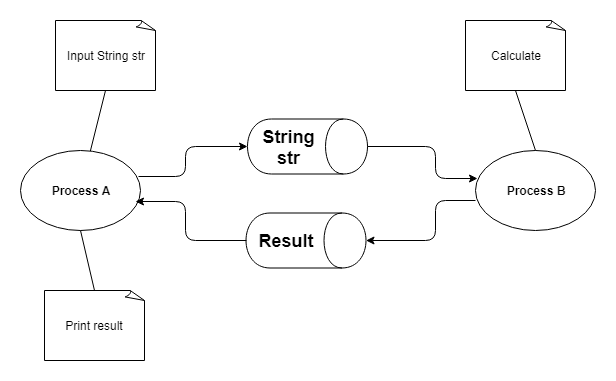
**ĐẦU RA:** Kết quả của phép tính nhập vào

Ví dụ với chuỗi str = “1+2+(2-3-4)-((3+4)-5)” thì ta sẽ có kết quả như sau.

**ĐẦU VÀO**: str = “1+2+(2-3-4)-((3+4)-5)”

**ĐẦU RA**: result = 0

Từ yêu cầu bài toán như trên, em tổ chức thành hai tiến trình như sau.



*Hình 1.2.1 - Sơ đồ mô phỏng bài toán giao tiếp hai chiều giữa tiến trình cha-con bằng hai đường ống.*

Các bước thực hiện của chương trình như sau:

* B1: Tiến trình A sẽ nhận vào là chuỗi str sau đó thông qua đường ống gửi đến tiến trình B để kiểm tra.
* B2: Tiến trình B sẽ nhận chuỗi str gửi đến từ tiến trình A và tiến hành xử lý tính toán. Sau đó tạo ra một báo cáo kết quả dạng chuỗi và thông qua một đường ống khác và gửi lại tiến trình A.
* B3: Tiến trình A sẽ nhận chuỗi báo cáo và in kết qua ra màn hình thông qua STD out.

Em sử dụng ngôn ngữ lập trình C trên hệ điều hành Linux để viết chương trình mô phỏng bài toán trên. Đối với ngôn ngữ C trên hệ điều hành Linux, để tạo ra một đường ống thì em sẽ làm như sau.

int fds[2];

pid\_t pid;

pipe(fds);

Với hàm pipe được cung cấp từ thư viện unistd.h. Kết quả trả về sẽ hai file descriptor của hai đầu đường ống. Nguyên tắc để giao tiếp qua đường ống là đầu đọc khi được mở ra thì sẽ chờ cho đến khi đầu còn lại ghi xong và đóng lại. Chính nhờ tín hiệu đóng của đầu kia mà đầu còn lại mới bắt đầu đọc dữ liệu.

Ngoài ra để tạo ra một tiến trình con trong ngôn ngữ C trên nền hệ điều hành Linux thì làm như sau.

pid\_t pid;

pid = fork();

if(pid == (pid\_t) 0)

{

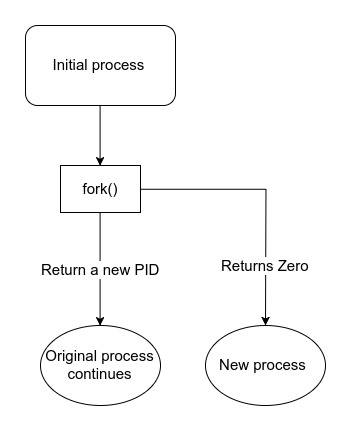
// Child process.

} else {

// Parent process.

}

Trong đó hàm folk khi tạo mới thành công tiến trình con sẽ trả về mã PID của tiến trình con, trong trường hợp bị lỗi sẽ trả về một số nhỏ hơn 0. PID sẽ nhận giá trị là 0 trong trường hợp đoạn chương trình chạy chính là phần thân của tiến trình con.



*Hình 1.2.2 - Sơ đồ mô phỏng quá trình tạo ra tiến trình con bằng hàm fork*

Tiếp theo là quá trình trao đổi đọc ghi dữ liệu bằng đường ống giữa hai tiến trình cha và con. Đầu tiên chúng ta phải xác định đâu là đầu để đọc và đâu là đầu để ghi. Mội quy tắc chung được dùng phổ biến là đầu 0 để đọc và đầu 1 để ghi.

Tiếp theo đó, trước khi ghi dữ liệu vào một đầu thì phải đóng đầu kia lại. Và tương tự trước khi đọc dữ liệu từ đầu kia ra thì cũng phải đóng đầu đọc lại. Khi mở một đầu ra để ghi hay đọc cũng phải được xác định thông qua tham số “r” là đọc hoặc “w” là ghi.

**\* Thuật toán tính giá trị biểu thức:** ở đây em dùng stack chuyển biểu thức trung tố (infix) sang biểu thức hậu tố (postfix) rồi tính toán trên đó. Chi tiết thuật toán được trình bày ở phần 2 phụ lục.

Từ cơ sở lý thuyết và các phân tích bằng mã lệnh C trên nền hệ điều hành Linux ở trên, em đã viết chương trình hoàn chỉnh.

**Mã lệnh đầy đủ của chương trình nằm ở phần 1 phụ lục.**

## **CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ**

**Môi trường làm việc:**

- Hệ điều hành Ubuntu

- Ngôn ngữ lập trình C

- Công cụ IDE Code Block

**3.1. CHẠY CHƯƠNG TRÌNH BÀI TOÁN GIAO TIẾP HAI CHIỀU GIỮA TIẾN TRÌNH CHA VÀ CON BẰNG HAI ĐƯỜNG ỐNG MỘT CHIỀU**

Từ mã lệnh ở phần phụ lục 1 trong báo cáo này, giả sử toàn bộ mã lệnh được lưu trong tệp tin **pipe.c**

* Trên hệ điều hành Linux chạy lên gcc để biên dịch chương trình:

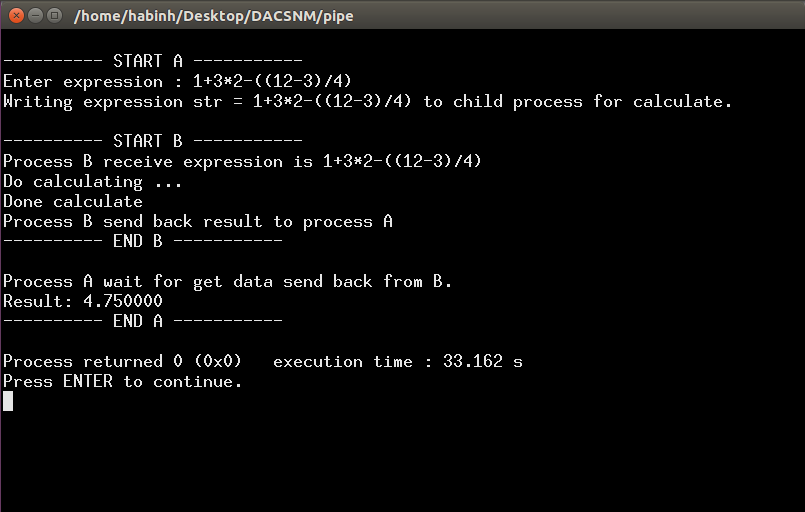
*gcc pipe.c -o pipe*

* Để thực thi chương trình thì chạy lệnh:

*./pipe*

* Người dùng nhập vào 1 chuỗi là biểu thức cần tính toán.

**Kết quả trả về sẽ như sau:**



*Hình 1.3.1 – Kết quả demo chương trình pipe.*

**Nhận xét kết quả:**

Bài toán trên với đầu vào biểu thức là chuỗi “1+3\*2-((12-3)/4)”. Đoạn chương trình sử dụng hai đường ống một chiều để giao tiếp, chính vì vậy mà sau khi gửi dữ liệu xong, tiến trình A sẽ phải chờ cho đến khi dữ liệu được trả về từ tiến trình B để ghi ra màn hình thì mới kết thúc được.

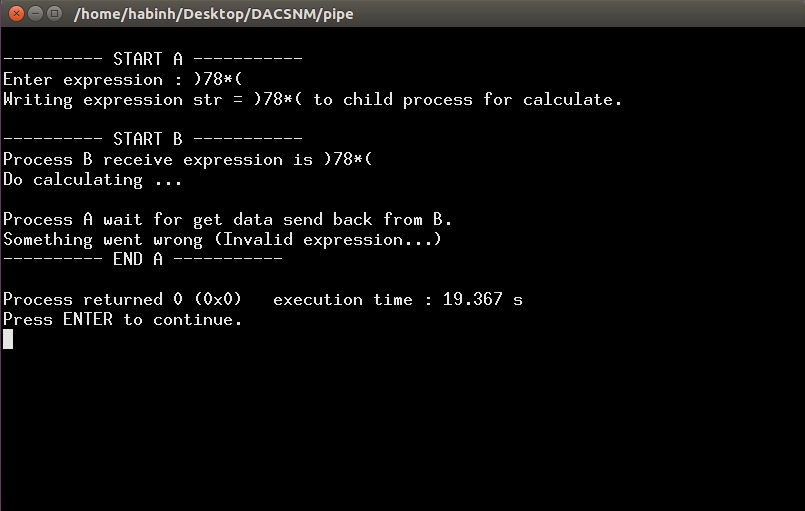
**3.2. ĐÁNH GIÁ**

**\* Ưu điểm:**

* Nắm được các kiến thức cơ bản và sử dụng được các hàm trong cơ chế giao tiếp các tiến trình bằng cơ chế đường ống pipe.
* Ứng dụng được để xây dựng chương trình giải quyết bài toán yêu cầu.

\* **Nhược điểm:**

* Chưa xây dựng được giao diện đẹp mắt và thân thiện với người dùng, chỉ dừng lại ở mức độ demo cho cơ chế giao tiếp giữa các tiến trình pipe.
* Exception ở quá trình tính toán chưa được xử lý thích hợp, exit tiến trình, gây khó chịu cho người dùng.



*Hình 1.3.2 – Kết quả demo chương trình pipe (Invalid Expression).*

## **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

* **Kết luận:**

Bài toán ở trên vẫn còn khá đơn giản, trên thực tế còn có các kiểu giao tiếp như là giao tiếp giữa một chuỗi các tiến trình nối tiếp nhau, đồng bộ một tập tiến trình con vào một tiến trình đầu ra cũng rất hay gặp trong thực tế. Hoặc bài toán nối tiến trình phức tạp không theo dạng nối tiếp mà theo cơ chế bất đồng bộ.

Đặc biệt là vấn đề về thời gian chạy của CPU khi một tiến trình làm việc quá dài thì thời gian chuyển qua lại giữa các tiến trình có thể anh hưởng đến dữ liệu gây mất mát hoặc sai sót, ảnh hưởng tới kết quả mong muốn của chương trình.

* Những vấn đề đã đạt được:

+ Trong quá trình làm đồ án, với những kiến thức đã học và tự tìm hiểu em đã hiểu hơn về các tiến trình và cơ chế giao tiếp giữa các tiến trình bằng đường ống PIPE.

+ Mặt khác em đã cải thiện được kĩ năng lập trình với ngôn ngữ C và kĩ năng sử dụng Code Block.

* Những vấn đề chưa đạt được:

+ Do thời gian có hạn và kiến thức của em còn hạn chế nên vẫn còn tồn tại một số vấn đề như giao diện chưa đẹp mắt, chưa test hết các exception cũng như bộ input lớn.

* **Hướng phát triển:**

- Tiếp tục nghiên cứu lý thuyết và thực hành.

- Khắc phục những vấn đề còn tồn tại chưa giải quyết được.

- Xây dựng thêm giao diện.

# PHẦN II: LẬP TRÌNH MẠNG

**TIÊU ĐỀ: Xây dựng chương trình xem điểm thi**

Yêu cầu đề tài:

Tìm hiểu và sử dụng kỹ thuật lập trình Socket xây dựng chương trình xem điểm thi theo mô hình Client - Server.

* Chương trình Client biểu diễn thông tin sinh viên, cho phép người dùng nhập mã số sinh viên gửi lên Server để xem điểm.
* Chương trình Server xử lý thông tin nhận được, truy vấn dữ liệu để trả về cho Client.

## **CHƯƠNG 1. CƠ SỞ LÝ THUYẾT**

Qua yêu cầu đề tài, thì đầu tiên em tự đặt ra các câu hỏi như sau và tìm cách trả lời những câu hỏi đó.

**Công nghệ và ngôn ngữ lập trình nào sẽ được sử dụng ?**

Trong đồ án này thì em quyết định sử dụng hệ quản trị cơ sở dữ liệu mySQL và ngôn ngữ lập trình Java để xây dựng chương trình.

Lý do chọn hai công nghệ này là vì Java có kiến trúc tổ chức mã lệnh tốt tuân thủ OOP, lại được sử dụng trong công việc và dạy học rất nhiều. Việc sử dụng Java để xây dựng khối Server, Client là phù hợp nhất với em. MySQL là một trong những cơ sở dữ liệu rất phổ biến hiện nay. Nó giàu các tính năng, cơ chế xử lý nhanh và ổn định cùng với sự tin cậy cao và dễ sử dụng, bắt đầu với MySQL là khá đơn giản và dễ dàng.

**Khối Server sẽ làm công việc gì ?**

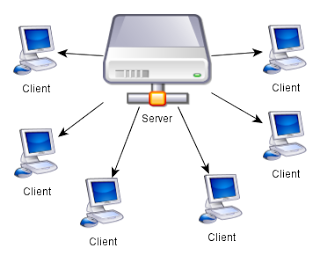
Khối Server viết bằng Java sẽ nhận các trách nhiệm sau:

1. Mở kết nối TCP tại một cổng nhất định để tiếp nhận các request gửi từ client lên và xử lý.
2. Làm việc với database, gửi response kết quả xử lý yêu cầu đó lại cho client tương ứng.

**Khối Client sẽ làm công việc gì ?**

Khối client sẽ đảm nhiệm các công việc sau:

1. Mở socket kết nối server để người dùng truy cập và trả về giao diện ứng dụng để người dùng tương tác.
2. Tiếp nhận yêu cầu từ người dùng rồi từ đó tạo kết nối TCP đến TCP Server để gửi dữ liệu rồi chờ nhận kết quả trả về.



*Hình 2.1.1 - Minh họa về hoạt động của cơ chế Client-Server*

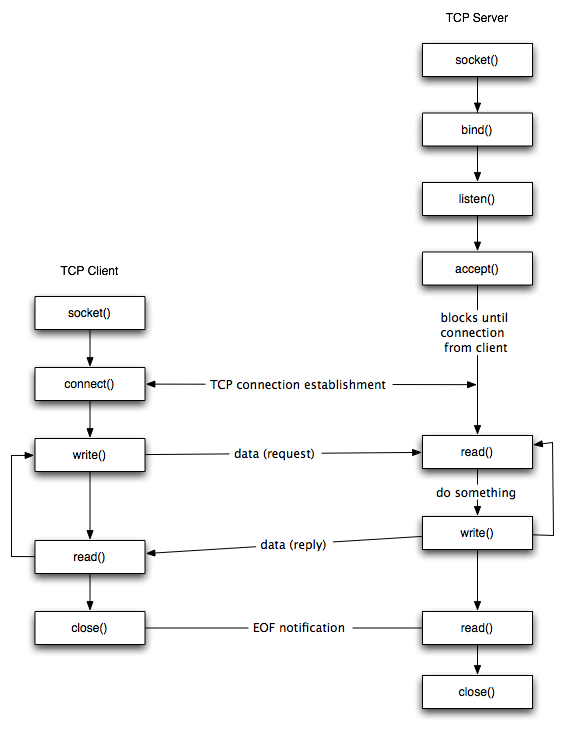
**Cơ sở lý thuyết về giao tiếp TCP/IP**

TCP – TCP là viết tắt của Transmission Control Protocol. TCP thường được sử dụng qua giao thức Internet (Internet Protocol), được gọi là TCP/IP. Giao thức này cho phép giao tiếp tin cậy giữa hai ứng dụng.

Socket cung cấp cơ chế truyền thông giữa hai máy tính sử dụng TCP. Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hai cổng (port). Thông qua các cổng này một quá trình có thể nhận và gởi dữ liệu với các quá trình khác.

Một máy khách (client) tạo ra socket để kết nối đến với máy chủ (server). Khi được chạy, server cần được xác định rõ địa chỉ IP và sẽ “lắng nghe” trên một port cụ thể. Server sẽ nằm trong trạng thái này cho đến khi client gửi đến một yêu cầu kết nối. Sau khi được server chấp nhận, một connection sẽ hình thành cho phép server và client giao tiếp với nhau.

Trong đồ án này, trọng tâm để giao tiếp giữa client và server là thông qua Socket TCP, và có thể mô tả thông qua mô hình dưới đây.



*Hình 2.1.2 - Sơ đồ quá trình trao đổi dữ liệu giữa Client và Server thông qua TCP*

**Các bước thiết lập kết nối TCP giữa hai máy tính sử dụng socket:**

1. Máy chủ khởi tạo một đối tượng ServerSocket với một cổng giao tiếp (port).
2. Máy chủ gọi phương thức accept() của lớp ServerSocket. Phương pháp này đợi cho đến khi một máy khách kết nối đến máy chủ trên cổng đã cho.
3. Trong khi máy chủ đang chờ đợi, một máy khách khởi tạo đối tượng Socket, xác định tên máy chủ (IP hoặc domain) và số cổng để kết nối.
4. Đối tượng socket của máy khách cố gắng kết nối máy khách tới máy chủ đã chỉ định và số cổng. Nếu truyền thông được thiết lập, máy khách bây giờ có một đối tượng socket có khả năng giao tiếp với máy chủ.
5. Ở phía máy chủ, phương thức accept() trả về một tham chiếu đến một socket mới trên máy chủ được kết nối với socket của máy khách.

Sau khi kết nối được thiết lập, máy chủ và máy khách có thể truyền và nhận thông tin thông qua OutputStream và InputStream.

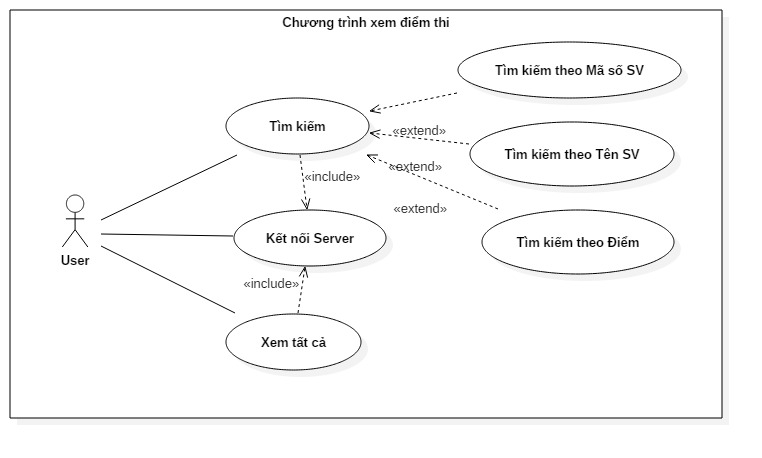
## **CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

* 1. **SƠ ĐỒ HOẠT ĐỘNG MÔ TẢ NGHIỆP VỤ HỆ THỐNG**

**Mô tả yêu cầu:**

Trong các kì thi quan trọng như tuyển sinh Đại học hay tuyển sinh THPT… việc tổng hợp điểm số của các thí sinh trong cơ sở dữ liệu và xây dựng hệ thống để tra cứu điểm là rất cần thiết. Trong đồ án Cơ sở ngành mạng (phần Lập trình mạng), em tiến hành xây dựng chương trình demo đơn giản để xử lý việc xem điểm thi này, thông qua đó để hiểu rõ hơn về mô hình Client-Server cũng như kỹ thuật lập trình Socket TCP.

**Sơ đồ usecase:**



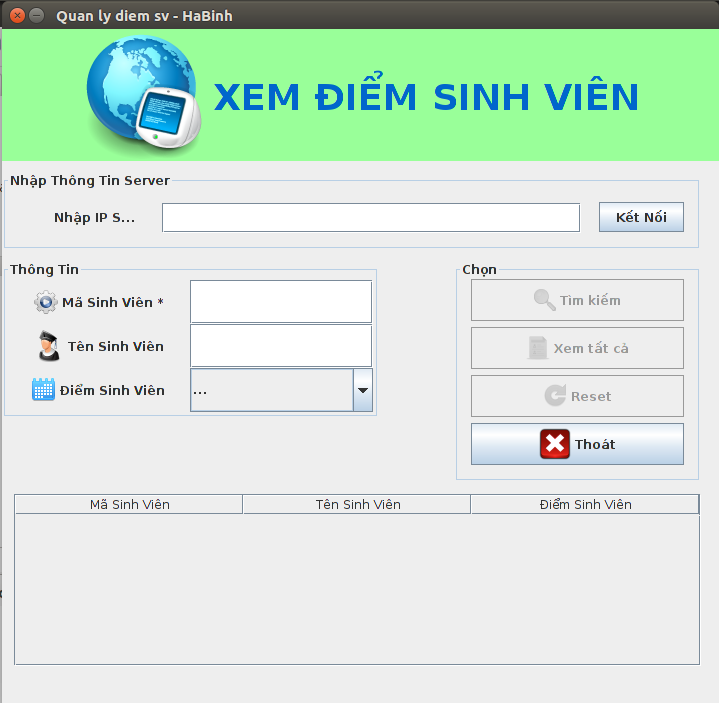
*Hình 2.2.1 - Sơ đồ ca sử dụng use-case*

*Luồng sự kiện chính*: Khi người dùng khởi động ứng dụng, hệ thống yêu cầu người dùng nhập vào địa chỉ IP của Server sau đó nhấn kết nối.

* Nếu kết nối thành công: Hiển thị thông báo và enable các chức năng Xem tất cả và Tìm kiếm thông tin.
* Nếu kết nối không thành công: Hiển thị thông báo yêu cầu người dùng kiểm tra và nhập lại.

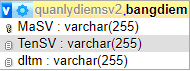
* 1. **THIẾT KẾ**

**Giao diện:**



*Hình 2.2.2 – Giao diện chính của chương trình*

**Cơ sở dữ liệu:**



*Bảng 2.2.1 –Bảng mô tả cơ sở dữ liệu*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Độ rộng** | **Mô tả** | **Chú thích** |
| 1 | MaSV | Varchar | 255 | Khóa chính | Mã số sinh viên |
| 2 | TenSV | Varchar | 255 |  | Họ tên sinh viên |
| 3 | dltm | Varchar | 255 |  | Điểm thi |

Từ cơ sở lý thuyết và các phân tích, thiết kế ở trên, em đã viết chương trình hoàn chỉnh bao gồm 2 phần Client và Server.

**Link source code của chương trình nằm ở phần 3 phụ lục.**

## **CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ**

**Môi trường làm việc:**

* Hệ điều hành Ubuntu
* Java 8
* Công cụ IDE Netbean, xampp

**3.1. CHẠY CHƯƠNG TRÌNH XEM ĐIỂM THI**

Tiến hành biên dịch và chạy chương trình bằng command line hoặc dùng IDE, ở đây em dùng IDE Netbean để thuận tiện cho việc thao tác cũng như viết mã, debug.

* Build and run Server

Xem thông báo trạng thái server ở console

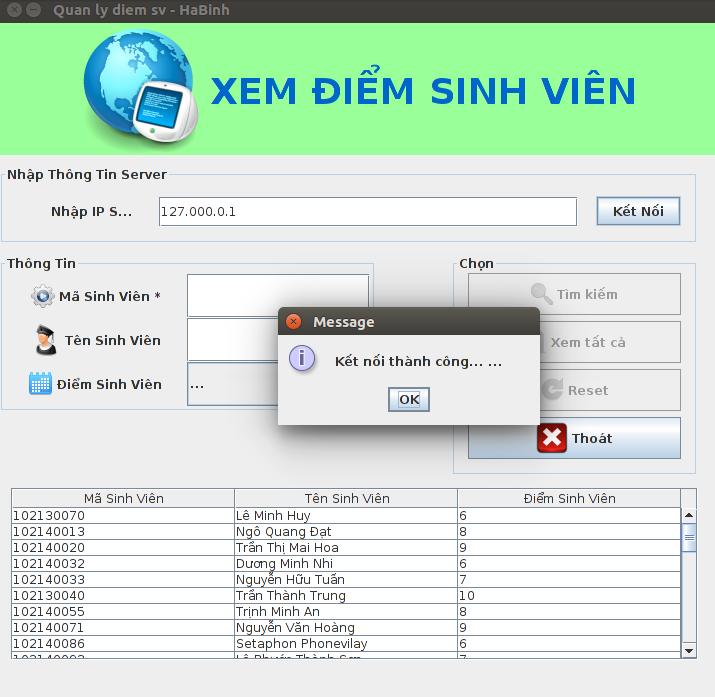
Kiểm tra địa chỉ ip của máy.

* Build and run Client

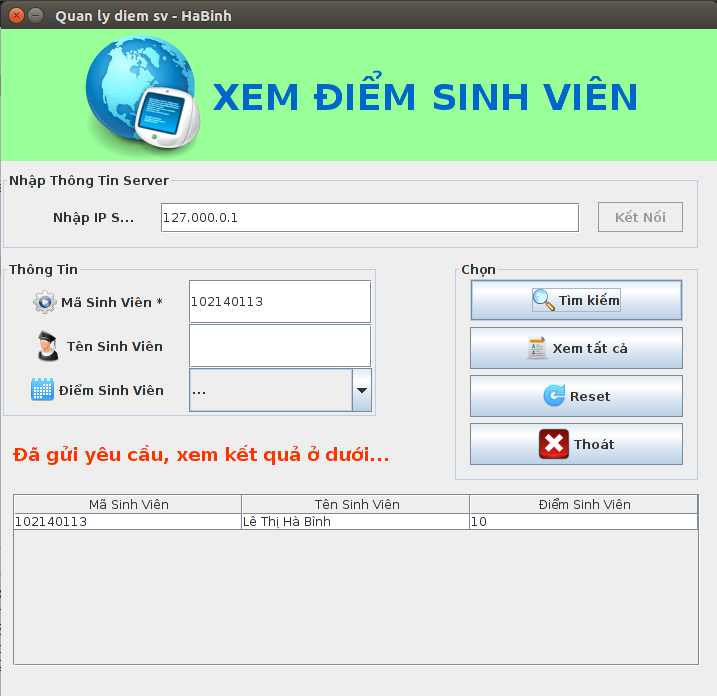
Nhập địa chỉ ip của Server (2 trường hợp: nhập đúng hoặc sai)

Khi đã kết nối được với server, tiến hành test các chức năng của hệ thống.

*\* Khởi động nhiều client và thực hiện các thao tác để kiểm tra socket và khả năng đáp ứng của server cũng như hệ thống.*



*Hình 2.3.1 – Kết quả demo chương trình xem điểm thi.*



*Hình 2.3.2 – Kết quả demo chức năng tìm kiếm.*

**\* *Hình ảnh khác của chương trình được trình bày trong phần 4 phụ lục***

**3.2. ĐÁNH GIÁ**

**\* Ưu điểm:**

* Nắm được các kiến thức cơ bản về socket TCP và mô hình Client server. Sử dụng được các kiến thức đó để lập trình.
* Xây dựng được chương trình giải quyết bài toán yêu cầu.
* Kết quả tìm kiếm chính xác và đúng yêu cầu.
* Thời gian đáp ứng nhanh.
* Thao tác dễ dàng cho người dùng.

\* **Nhược điểm:**

* Giao diện chưa thật sự đẹp mắt.
* Chỉ dừng lại ở mức độ demo phục vụ cho đồ án nên các chưa năng chưa đầy đủ như trong thực tế.

## **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

* **Kết luận:**

Qua đồ án lập trình mạng này, em học được rất nhiều kiến thức, trong đó những phần quan trọng nhất có thể kể ra dưới đây đó là:

1. Cách thức vận hành mô hình TCP/IP và thực hành truyền nhận dữ liệu giữa Client, Server thông qua TCP.
2. Kỹ năng phân tích, thiết kế và xây dựng một ứng dụng.
3. Rèn luyện kỹ năng lập trình công nghệ Java.
4. Kỹ năng triển khai ứng dụng và vận hành ứng dụng.

* **Hướng phát triển:**

Dự định sắp tới của em trong việc mở rộng dự án này đó là:

* Xây dựng lại giao việc ở client đẹp và tăng tính tương tác lên nữa.
* Xây dựng thêm các chức năng bộ lọc để tìm kiếm nâng cao.
* Tiến hành phân trang khi query dữ liệu để chương trình hoạt động tốt với cơ sở dữ liệu lớn.

# PHỤ LỤC

**Phần 1. MÃ NGUỒN BÀI TOÁN GIAO TIẾP GIỮA CÁC TIẾN TRÌNH**

#include <stdio.h>

#include <unistd.h>

#include<stdio.h>

#define SIZE 1024 \* 10

#define MAX 4000

// STACK

typedef struct stack\_t{

char elements[MAX];

int top;

} stack;

int isEmpty(stack S){

return S.top<=-1;

}

int isFull(stack S){

return S.top >= MAX-1;

}

void create(stack \*S){

S->top=-1;

}

void push(stack \*S, char a){

if(!isFull(\*S)){

S->elements[++S->top] = a;

}

}

char pop(stack \*S){

if(!isEmpty(\*S)){

return S->elements[S->top--];

}

return 0;

}

char peek(stack \*S){

if(!isEmpty(\*S)){

return S->elements[S->top];

}

return 0;

}

void print(stack S){

int i;

for(i=0;i<=S.top;i++){

printf("%c ",S.elements[i]);

}

}

// END STACK

// Thu tu uu tien cac toan tu

int ktuutien(char a)

{

if (a == '-')

return 0;

if (a == '+')

return 0;

if (a == '\*')

return 1;

if (a == '/')

return 1;

}

int kiemtraso(char a)

{

if ((a >= '0' && a <= '9') || a == '.')

return 1;

return 0;

}

int kttoantu(char ch)

{

if (ch == '+' || ch == '-' || ch == '\*' || ch == '/')

return 1;

return 0;

}

int toanhang(char a)

{

if (a >= 48 && a <= 57)

return 1;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ham` chuyen tu 1 bieu thuc trung to--->hau to \*\*\*\*\*\*\*\*\*\*\*\*\*/

int conver(char a[], char b[])

{

stack k;

create(&k);

int dem = 0;

for (int i = 0; i < strlen(a); i++)

{

if (kttoantu(a[i]) && kttoantu(a[i + 1])) ///kiem tra du lieu dau vao`

return 0;

if ((a[i] >= 97 && a[i] <= 122) || (a[i] >= 65 && a[i] <= 90))

return 0;

}

for (int i = 0; i < strlen(a); i++)

{

char ch = a[i];

if (kiemtraso(ch))

{

b[dem++] = ch;

if (kttoantu(a[i + 1]))

b[dem++] = ' ';

//dau cach dung de Phan biet giua cac so co nhieu` chu so voi nhau

}

if (ch == '(')

push(&k,ch);

if (kttoantu(ch))

{

if (!isEmpty(k))

{

if ((kttoantu(peek(&k))) && (ktuutien(ch) == ktuutien(peek(&k)) || ktuutien(ch) < ktuutien(peek(&k)))) // cau lenh nay` kiem tra do uu tien giua cac toan tu

{

char m;

m = pop(&k);

b[dem++] = m;

push(&k,ch);

}

else

{

push(&k,ch);

}

}

else

push(&k,ch);

}

if (ch == ')') // neu Phat hien dau ) thi ta pop cho toi khi gap dau '(" thi` dung`

{

char d;

while (1)

{

d=pop(&k);

if (d == '(')

break;

b[dem++] = d;

}

}

}

// lay toan tu con lai trong stack...

while (!isEmpty(k))

{

char m;

char n;

m=pop(&k);

b[dem++] = m;

}

b[dem] = '\0';

}

float doi(char a)

{

switch (a)

{

case '1': return 1;

case '2': return 2;

case '3': return 3;

case '4': return 4;

case '5': return 5;

case '6': return 6;

case '7': return 7;

case '8': return 8;

case '9': return 9;

case '0': return 0;

}

}

//// ham` bien doi chuoi sang dang so

float biendoi(char a[])

{

float d = 0;

int n = strlen(a);

if (strlen(a))

{

for (int i = 0; i < n; i++)

{

if (a[i] == '.')

{

int dem = 1;

float bd1 = 1, bd = 0;

for (int j = i + 1; j < n; j++)

{

for (int i = 0; i < dem; i++)

bd1 = bd1 \* 10;

bd = bd + doi(a[j]) / bd1;

}

d = d + bd;

break;

}

else

d = d \* 10 + doi(a[i]);

}

return d;

}

return 0;

}

// Tinh toan bieu thuc hau to

float tinhtoan(char pos[SIZE])

{

char ch = '\0';

int dem1 = 0;

float data[1000000];

char postfix[10000];

conver(pos,postfix);

pos=postfix;

int n = strlen(pos);

for (int i = 0; i < n;)

{

char b[100];

int dem = 0;

while (pos[i] != ' '&&i < n&&!kttoantu(pos[i]))

{

b[dem++] = pos[i++];

}

if (dem)

{

b[dem] = '\0';

data[dem1++] = biendoi(b);

}

if (kttoantu(pos[i]))

{

switch (pos[i])

{

case '+':

{

float temp = data[dem1 - 1] + data[dem1 - 2];

data[dem1 - 2] = temp;

dem1--;

}break;

case '-':

{

float temp = data[dem1 - 2] - data[dem1 - 1];

data[dem1 - 2] = temp;

dem1--;

}break;

case '\*':

{

float temp = data[dem1 - 1] \* data[dem1 - 2];

data[dem1 - 2] = temp;

dem1--;

}

break;

case '/':

{

float temp = data[dem1 - 2] / data[dem1 - 1];

data[dem1 - 2] = temp;

dem1--;

}break;

}

}

i++;

}

return data[0];

}

void reader(FILE\* stream)

{

char buffer[SIZE];

if(!feof(stream) && !ferror(stream) && fgets(buffer,sizeof(buffer),stream) != NULL) {

fputs(buffer,stdout);

} else {

fputs("Something went wrong (Invalid expression...)\n",stdout);

}

}

int main()

{

int fds\_1[2]; //pipe 1 (result)

int fds\_2[2]; //pipe 2 (inputString)

pid\_t pid;

pipe(fds\_1);

pipe(fds\_2);

// Read 0, Write 1

//ParentRead fds\_1[0]

//ParentWrite fds\_2[1]

//ChildRead fds\_2[0]

//ChildWrite fds\_1[1]

pid = fork();

if(pid == (pid\_t) 0)

{

/\* Child \*/

sleep(1);

FILE\* stream;

close(fds\_2[1]);

stream = fdopen(fds\_2[0], "r");

char buffer[SIZE];

fgets(buffer,sizeof(buffer),stream);

if(sizeof(buffer)>0){

printf("\n%s\n", "---------- START B -----------");

}

char strReceive[SIZE];

char result[SIZE] = "Result: ";

char tmpStr[SIZE];

// Read inputString from parent

sscanf(buffer, "%s", &strReceive);

printf("Process B receive expression is %s\n", strReceive);

close(fds\_2[0]);

printf("%s\n", "Do calculating ...");

sprintf(tmpStr, "%3.3f", tinhtoan(strReceive)); //tmpStr = tinhtoan(strReceive)

strcat(result, tmpStr); //result += tmpStr

printf("%s\n", "Done calculate");

printf("%s\n", "Process B send back result to process A");

// Write back data (result) to parent.

close(fds\_1[0]);

stream = fdopen(fds\_1[1], "w");

fprintf(stream, "%s\n", result);

fflush(stream);

printf("%s\n", "---------- END B -----------");

close(fds\_1[1]);

}

else

{

/\* Parent \*/

FILE\* stream;

close(fds\_2[0]);

stream = fdopen(fds\_2[1], "w");

// The string (inputString) write to child process for calculate.

printf("\n%s\n", "---------- START A -----------");

char inputString[SIZE];

printf("Enter expression : ");

fflush(stdin);

gets(inputString);

printf("Writing expression str = %s to child process for calculate.\n", inputString);

fprintf(stream, "%s\n", inputString);

fflush(stream);

close(fds\_2[1]);

sleep(1);

// Get back data (result) send from child.

close(fds\_1[1]);

stream = fdopen(fds\_1[0], "r");

printf("\n%s\n", "Process A wait for get data send back from B.");

reader(stream);

printf("%s\n", "---------- END A -----------");

close(fds\_1[0]);

}

return 0;

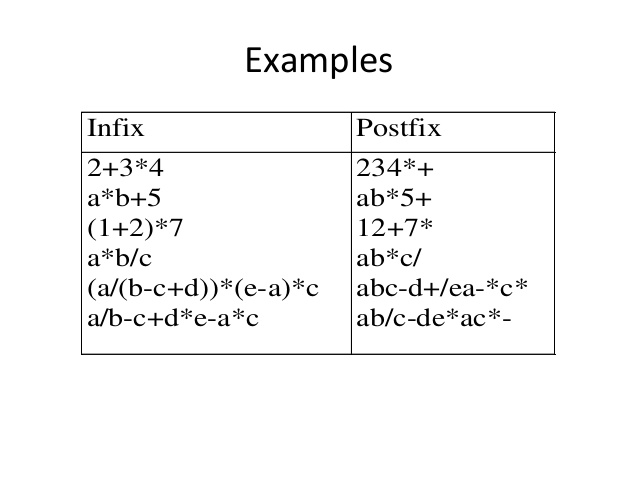
}

**Phần 2. THUẬT TOÁN TÍNH GIÁ TRỊ BIỂU THỨC**

**\* Thuật toán chuyển biểu thức trung tố sang hậu tố bằng Stack**

Input: Infix là biểu thức trung tố.

Output: Postfix là biểu thức hậu tố.



1. Khởi tạo Stack rỗng.

2. Khởi tạo 2 chuỗi x và token; i, j lần lượt là index của Infix và Postfix.

3. Duyệt vòng lặp for từ i = 1 cho đến cuối chuỗi Infix:

* Nếu Infix[i] là toán hạng thì đưa vào Postfix.
* Nếu Infix[i] là toán tử thì kiểm tra độ ưu tiên
  + Nếu độ ưu tiên lớn hơn hoặc bằng phần tử đỉnh ngăn xếp thì Pop ngăn xếp S (lấy giá trị trên đỉnh của S) sau đó đưa vào Postfix.
  + Push Infix[i] vào ngăn xếp S.
* Nếu Infix[i] là “)” thì Pop ngăn xếp S (lấy giá trị trên đỉnh của S) sau đó đưa vào Postfix.

**\* Tính giá trị biểu thức hậu tố:**

Duyệt biểu thức dạng chuỗi từ trái sang phải:

Dùng hàm isdigit để kiểm tra:

+ Nếu là toán hạng thì dùng Push() đưa vào ngăn xếp S.

+ Nếu là toán tử thì Pop() 2 toán hạng trong ngăn xếp S ra, sau đó tính toán giá trị của chúng dựa vào toán tử này, sau đó Push() lại vào S.

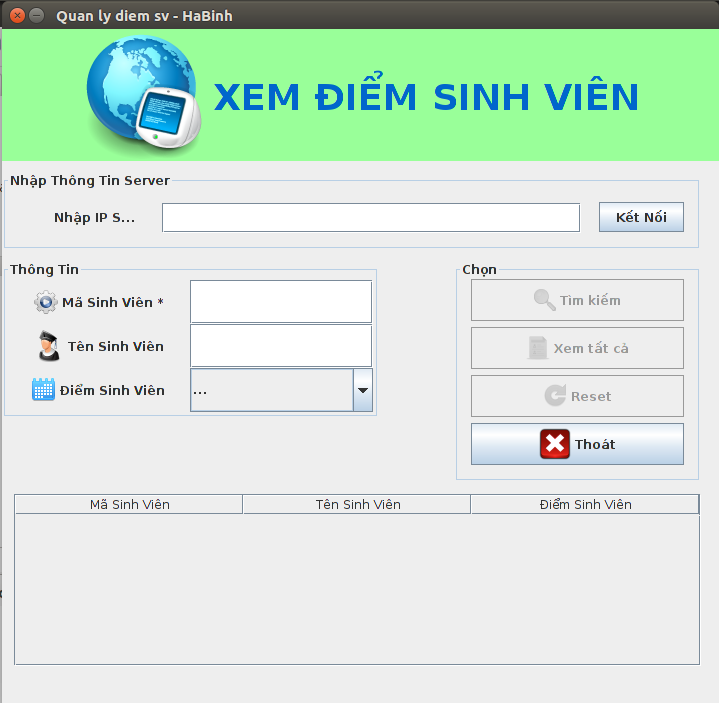
+ Thực hiện cho đến khi gặp kí tự "\0" kết thúc chuỗi.

+ Kết quả của biểu thức chính là phần tử còn lại cuối cùng trong ngăn xếp S.

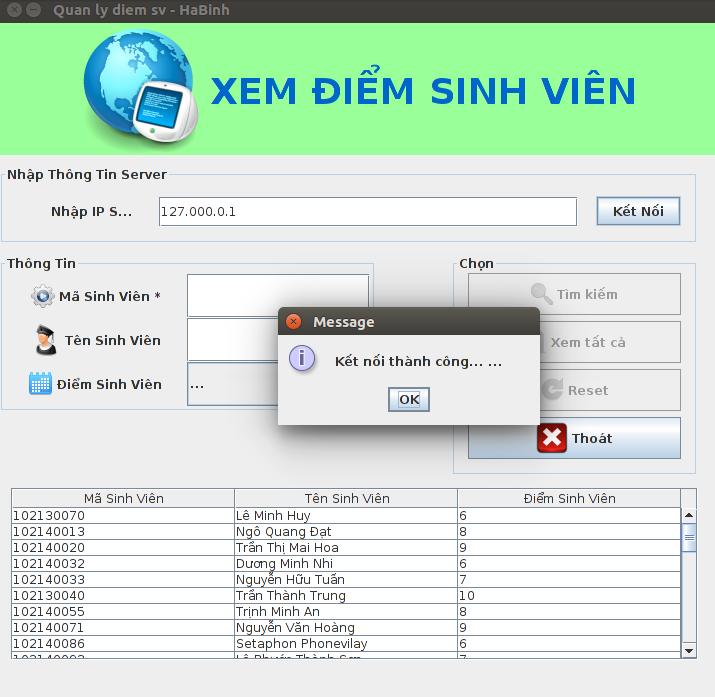
**Phần 3. LINK SOURCE CODE CHƯƠNG TRÌNH XEM ĐIỂM THI**

[*https://github.com/HaBinh/Xem-diem-sv*](https://github.com/HaBinh/Xem-diem-sv)

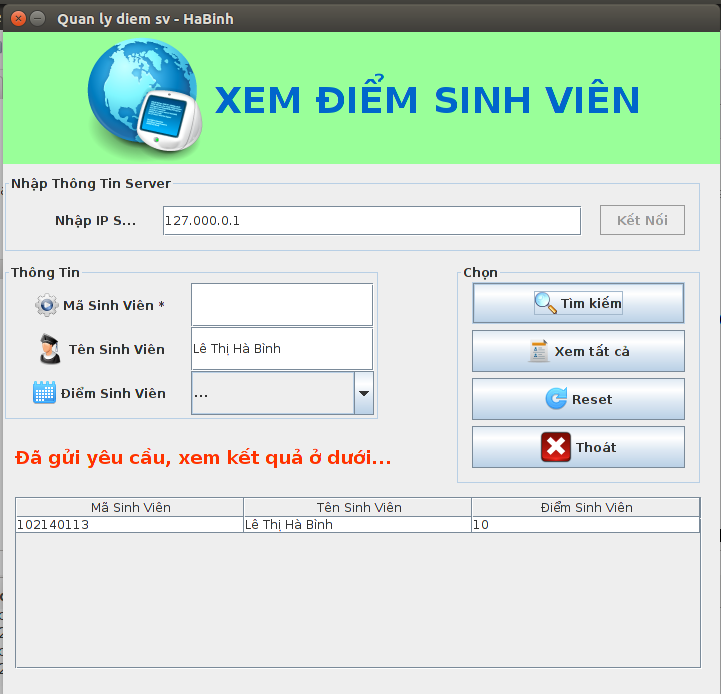
**Phần 4. HÌNH ẢNH CHƯƠNG TRÌNH XEM ĐIỂM THI**



*Hình 1 – Màn hình khi khởi động chương trình.*



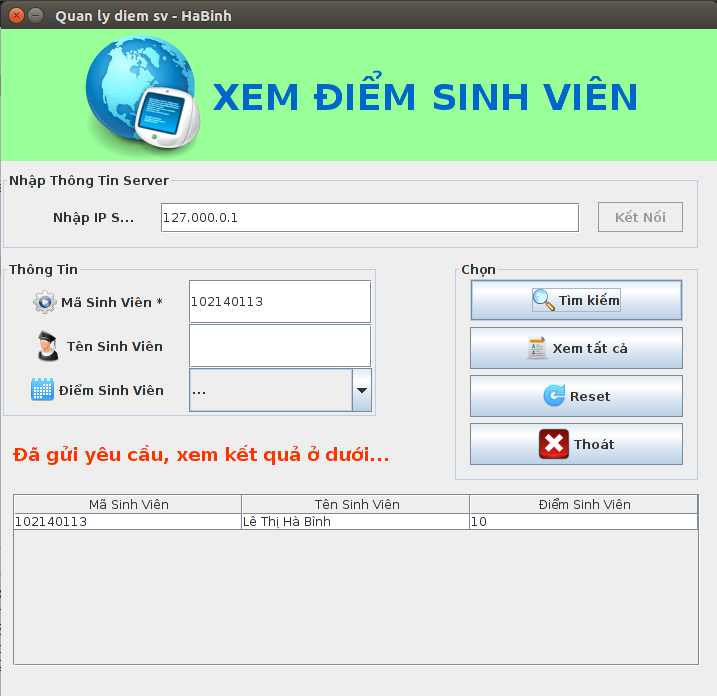
*Hình 2 – Kết nối thành công.*



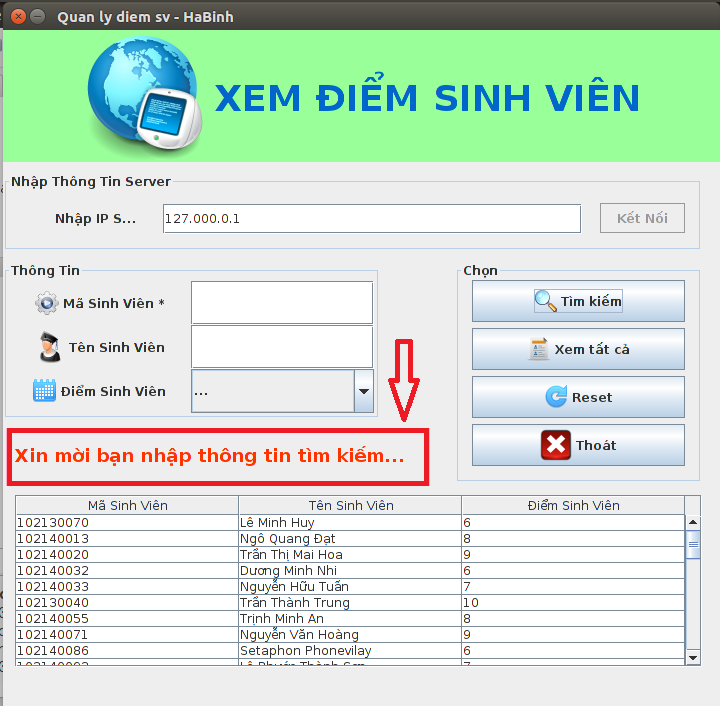
*Hình 3 – Tìm kiếm theo tên.*



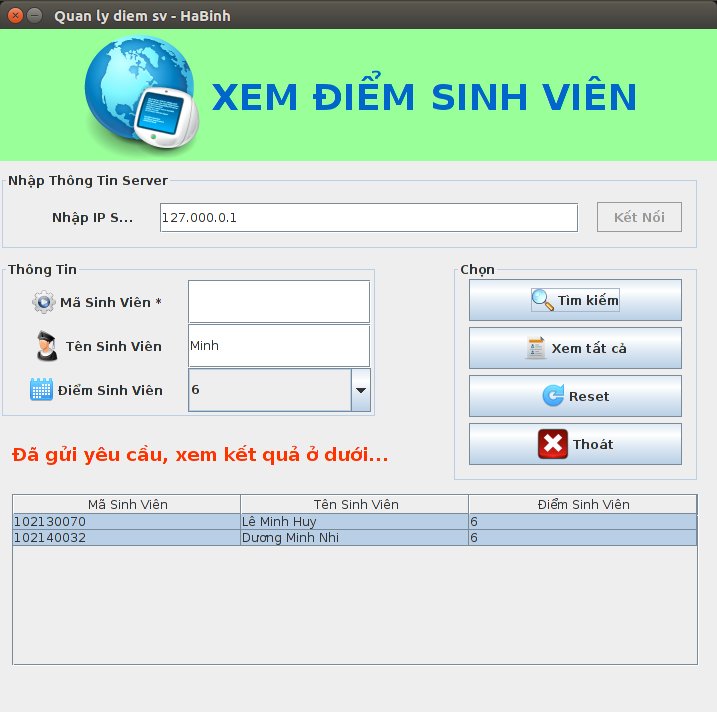
*Hình 4 – Tìm kiếm ra nhiều kết quả*



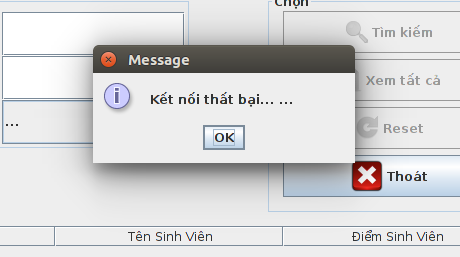
*Hình 5 – Tìm kiếm theo Mã SV*

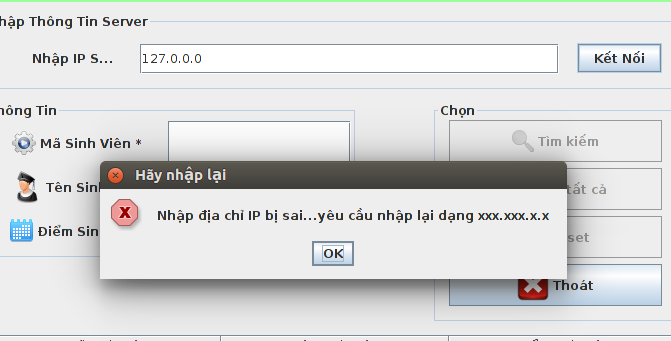


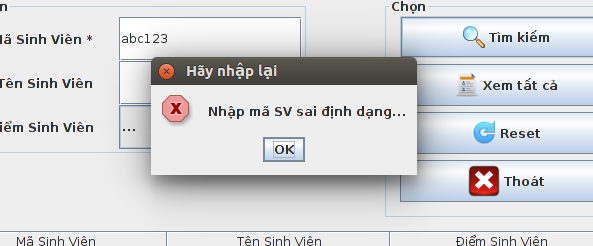
*Hình 6 – Không nhập thông tin tìm kiếm*



*Hình 7 – Tìm kiếm theo nhiều tiêu chí.*







*Hình 8 – Một số thông báo lỗi.*

# KẾT LUẬN CHUNG

Qua đồ án cơ sở ngành mạng này bọn em học và rút ra rất nhiều kiến thức bổ ích. Hiểu sâu hơn về nguyên lý vận hành và cách thức giao tiếp giữa các tiến trình, đặc biệt là cơ chế giao tiếp thông qua TCP/IP. Vận dụng được kiến thức học được vào việc tạo ra sản phẩm, thông qua đó kiến thức được củng cố tốt hơn. Về tiến độ công việc được giao, em cũng cơ bản hoàn thiện được đúng theo yêu cầu.

Em xin một lần nữa gửi lời cảm ơn chân thành đến thầy Nguyễn Văn Nguyên đã hướng dẫn tận tình để cho em hoàn thiện đồ án này.

*Xin chân thành cảm ơn.*

*Lê Thị Hà Bình*

**PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN**