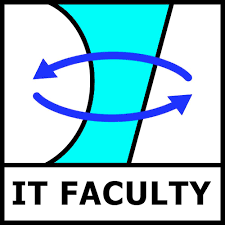
**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----------------------------------------------**



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

***ĐỀ TÀI***

**Giao tiếp giữa các tiến trình bằng cơ chế đường ống pipe.**

**Tìm hiểu giao thức SMTP, POP3 xây dựng chương trình mail client.**

**GVHD: ThS. Nguyễn Thị Lệ Quyên**

**SVTH : Trần Hữu Hồng Sơn**

**MSSV : 102160063**

***Đà Nẵng, 2019***

**MỞ ĐẦU**

Đồ án cơ sở ngành mạng là một trong những đồ án cơ sở của ngành học Công nghệ thông tin. Thông qua đồ án này, sinh viên có thể tìm hiểu được những kiến thức cơ bản nhất và có được nền tảng để có thể tự nghiên cứu nhiều hơn về những kiến thức liên quan đến hai bộ môn *Lập trình mạng và Nguyên lí hệ điều hành.*

Trong đồ án lần 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), với phần *Lập trình mạng*, em chọn đề tài tìm hiểu giao thức SMTP, POP3 để xây dựng chương trình demo ứng dụng mail-client. Với sự giảng dạy và hỗ trợ của các thầy cô bộ môn mạng cùng sự nỗ lực cá nhân, đồ án cơ sở ngành mạng đã được hoàn thành đúng với thời gian được giao.

Đặc biệt em xin trân trọng cảm ơn thầy Mai Văn Hà và cô Nguyễn Lê Thủy Tiên với những kiến thức học được từ học phần *Lập trình mạng và Nguyên lí hệ điều hành,* cảm cô Nguyễn Thị Lê Quyên vì những định hướng và nhận xét trong quá trình thực hiện đồ án này.

***Sinh viên thực hiện***

Trần Hữu Hồng Sơn

MỤC LỤC

[**MỞ ĐẦU 1**](#_Toc27659220)

[**MỤC LỤC 2**](#_Toc27659221)

[**PHẦN 1: NGUYÊN LÝ HỆ ĐIỀU HÀNH 3**](#_Toc27659222)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 4](#_Toc27659223)

[**1. Tiến trình trong Linux 4**](#_Toc27659224)

[**2. Các cơ chế giao tiếp các tiến trình 5**](#_Toc27659225)

[**3. Cơ chế giao tiếp các tiến trình bằng đường ống pipe 6**](#_Toc27659226)

[**4. Đường ống giao tiếp dữ liệu hai chiều 6**](#_Toc27659227)

[**5. Tạo đường ống 6**](#_Toc27659228)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ 7](#_Toc27659229)

[**1. Phân tích và thiết kế chương trình 7**](#_Toc27659230)

[**2. Thuật toán 9**](#_Toc27659231)

[CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ 10](#_Toc27659232)

[**1. Biên dịch và thực thi 10**](#_Toc27659233)

[**2. Đánh giá 11**](#_Toc27659234)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 12](#_Toc27659235)

[**1. Kết luận 12**](#_Toc27659236)

[**2. Hướng phát triển 12**](#_Toc27659237)

[**PHẦN 2: LẬP TRÌNH MẠNG 14**](#_Toc27659238)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 15](#_Toc27659239)

[**1. Các giao thức gửi/nhận mail 15**](#_Toc27659240)

[**2. JavaMail API 17**](#_Toc27659241)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ 18](#_Toc27659242)

[**1. Phân tích chức năng 18**](#_Toc27659243)

[**2. Thiết kế 19**](#_Toc27659244)

[CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ 21](#_Toc27659245)

[**1. Môi trường phát triển 21**](#_Toc27659246)

[**2. Demo chương trình 21**](#_Toc27659247)

[**3. Đánh giá: 27**](#_Toc27659248)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 27](#_Toc27659249)

[**1. Kết luận 27**](#_Toc27659250)

[**2. Hướng phát triển 28**](#_Toc27659251)

[**TÀI LIỆU THAM KHẢO 29**](#_Toc27659252)

# PHẦN 1: NGUYÊN LÝ HỆ ĐIỀU HÀNH

**ĐỀ TÀI: GIAO TIẾP GIỮA CÁC TIẾN TRÌNH**

**BẰNG CƠ CHẾ ĐƯỜNG ỐNG PIPE**

**Yêu cầu đề tài**

* Tìm hiểu cơ chế giao tiếp giữa các tiến trình sử dụng đường ống pipe.Đường ống giao tiếp trao đổi dữ liệu hai chiều.
* Xây dựng chương trình mô phỏng tính toán kết hợp sử dụng giao tiếp tiến trình hai chiều
* Tìm hiểu thuật toán ký pháp Balan để tính toán phép tính.

## CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. **Tiến trình trong Linux**

- Một trong những đặc điểm nổi bật của Linux là khả năng chạy đồng thời nhiều chương trình. Hệ điều hành xem mỗi đơn thể mã lệnh mà nó điều khiển là tiến trình (process). Một chương trình có thể bao gồm nhiều tiến trình kết hợp với nhau.

- Tiến trình là một thực thể điều khiển đoạn mã lệnh có riêng một không gian địa chỉ, có ngăn xếp stack riêng rẽ, có bảng chứa các thông số mô tả file được mở cùng tiến trình và đặc biệt có một định danh PID (Process Identify) duy nhất trong toàn bộ hệ thống vào thời điểm tiến trình đang chạy. Như chúng ta đã thấy, tiến trình không phải là một chương trình (tuy đôi lúc một chương trình đơn giản chỉ cần một tiến trình duy nhất để hoàn thành tác vụ, trong trường hợp này thì chúng ta có thể xem tiến trình và chương trình là một). Rất nhiều tiến trình có thể thực thi trên cùng một máy với cùng một Hệ điều hành, cùng một người dùng hoặc nhiều người dùng đăng nhập khác nhau.

- Để hoàn thành tác vụ của mình, một tiến trình có thể cần đến một số tài nguyên như CPU, bộ nhớ, các tập tin và thiết bị nhập/xuất.

- Các trạng thái của tiến trình:

* Đang chạy (running) : đây là lúc tiến trình chiếm quyền xử lý CPU dùng để tính toán hay thực các công việc của mình.
* Chờ (waiting) : tiến trình bị hệ điều hành tước quyền xử lý CPU, và chờ đến lượt cấp phát khác.
* Tạm dừng (suspend) : Hệ điều hành tạm dừng tiến trình. Tiến trình được đưa vào trạng thái ngủ (sleep). Khi cần thiết và có nhu cầu, Hệ điều hành sẽ đánh thức (wake up) hay nạp lại mã lệnh của tiến trình vào bộ nhớ. Cấp phát tài nguyên CPU để tiến trình tiếp tục hoạt động.

- 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 tiến trình.
* Định danh của tiến trình.
* 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).

- 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. **Các cơ chế giao tiếp các tiến trình**

- Đối với hệ điều hành, các tiến trình cùng hoạt động chia sẻ tốc độ xử lý của CPU, cùng dùng chung vùng nhớ và tài nguyên hệ thống khác. Các tiến trình được điều phối xoay vòng bởi hệ điều hành. Một chương trình của chúng ta nếu mở rộng dần ra, sẽ có lúc cần phải tách ra thành nhiều tiến trình để xử lý những công việc độc lập với nhau. Các lệnh của Linux thực tế là những lệnh riêng lẻ có khả năng kết hợp và truyền dữ liệu cho nhau thông qua các cơ chế như : đường ống pipe, chuyển hướng xuất nhập (redirect), phát sinh tín hiệu (signal),… Chúng được gọi là cơ chế giao tiếp liên tiến trình (IPC – Inter Process Comunication). Đối với tiến trình, chúng ta tìm hiểu cách tạo, hủy, tạm dừng tiến trình, đồng bộ hóa tiến trình và giao tiếp giữa các tiến trình với nhau.

Linux cung cấp một số cơ chế giao tiếp giữa các tiến trình gọi là IPC (Inter-Process Communication):

* Trao đổi bằng tín hiệu (signals handling)
* Trao đổi bằng cơ chế đường ống (pipe)
* Trao đổi thông qua hàng đợi thông điệp (message queue)
* Trao đổi bằng phân đoạn nhớ chung (shared memory segment)
* Giao tiếp đồng bộ dùng semaphore
* Giao tiếp thông qua socket

1. **Cơ chế giao tiếp các tiến trình bằng đường ống pipe**

- Một trong những cơ chế IPC cơ bản nhất là đường ống (pipe), tượng trưng cho dữ liệu tuần tự giữa các quá trình trong một đường ống.

- 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 thiết bị truyền thông tuần tự, dữ liệu có thể đọc (read) từ pipe cùng lúc ghi (write) lên pipe.

- Trên đường ống dữ liệu di chuyển 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 (input) của tiến trình này được chuyển đến làm dữ liệu nhập (input) cho tiến trình kia dưới dạng một dòng các byts.

- 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 và ngược lại. Thứ tự dữ liệu truyền qua pipe được bảo toàn theo nguyên tắc FIFO.

1. **Đường ống giao tiếp 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. Các tiến trình dùng cùng một đường ống để đọc và một đường ống để ghi. Tuy nhiên điều này cũng dễ gây ra tình trạng tắc nghẽn.

- Cả hai đường ống đều rỗng nếu đường ống rỗng hàm read() sẽ block cho đến khi có dữ liệu đổ vào hoặc khi đường ống bị đóng bởi bên ghi.

- Cả hai tiến trình cùng ghi dữ liệu: vùng đệm của một đường ống bị đầy, hàm write() sẽ block cho đến khi dữ liệu được lấy bớt ra từ một thao tác đọc read().

1. **Tạo đường ống**

Đối với lập trình C trên Linux, để tạo ra một đường ống ta làm như sau.

int data\_pipes[2];

int pid;

pipe(data\_pipes);

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 Linux ta làm như sau.

int pid;

pid = fork();

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

{

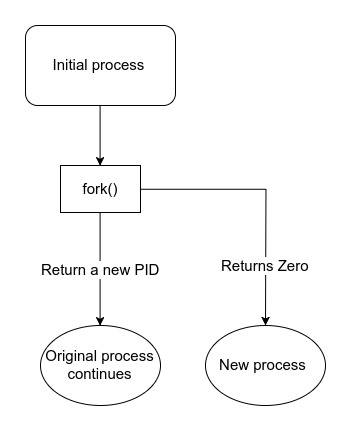
// do child process.

} else {

// do 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.



1. Sơ đồ mô phỏng quá trình tạo ra tiến trình con bằng hàm fork

Quá trình trao đổi đọc ghi dữ liệu bằng đường ống giữa hai tiến trình cha và con chúng ta cần 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 tương ứng với data\_pipes[0] và data\_pipes[1]

## CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ

1. **Phân tích và thiết kế chương trình**

- Chương trình sử dụng 2 tiến trình (một tiến trình con và một tiến trình cha) và 1 đường ống (pipe) bao gồm các bước sau:

* Tiến trình cha cho phép người dùng nhập vào từ bàn phím một chuỗi biểu diễn các phép tính như +, -, \*, /, sin, cos, tan, asin, acos, atan, ^, sqrt và ghi vào để tiến trình con nhận dữ liệu.
* Tiến trình con nhận dữ liệu đầu vào, sau đó tính toán sử dụng thuật toán Ký pháp Balan.
* Sau khi tính toán tiến trình con ghi ra kết quả.
* Tiến trình cha nhận kết quả sau khi đã được tính toán ở tiến trình con và in ra màn hình.

**Đầu vào:** Chuỗi biểu diễn phép tính.

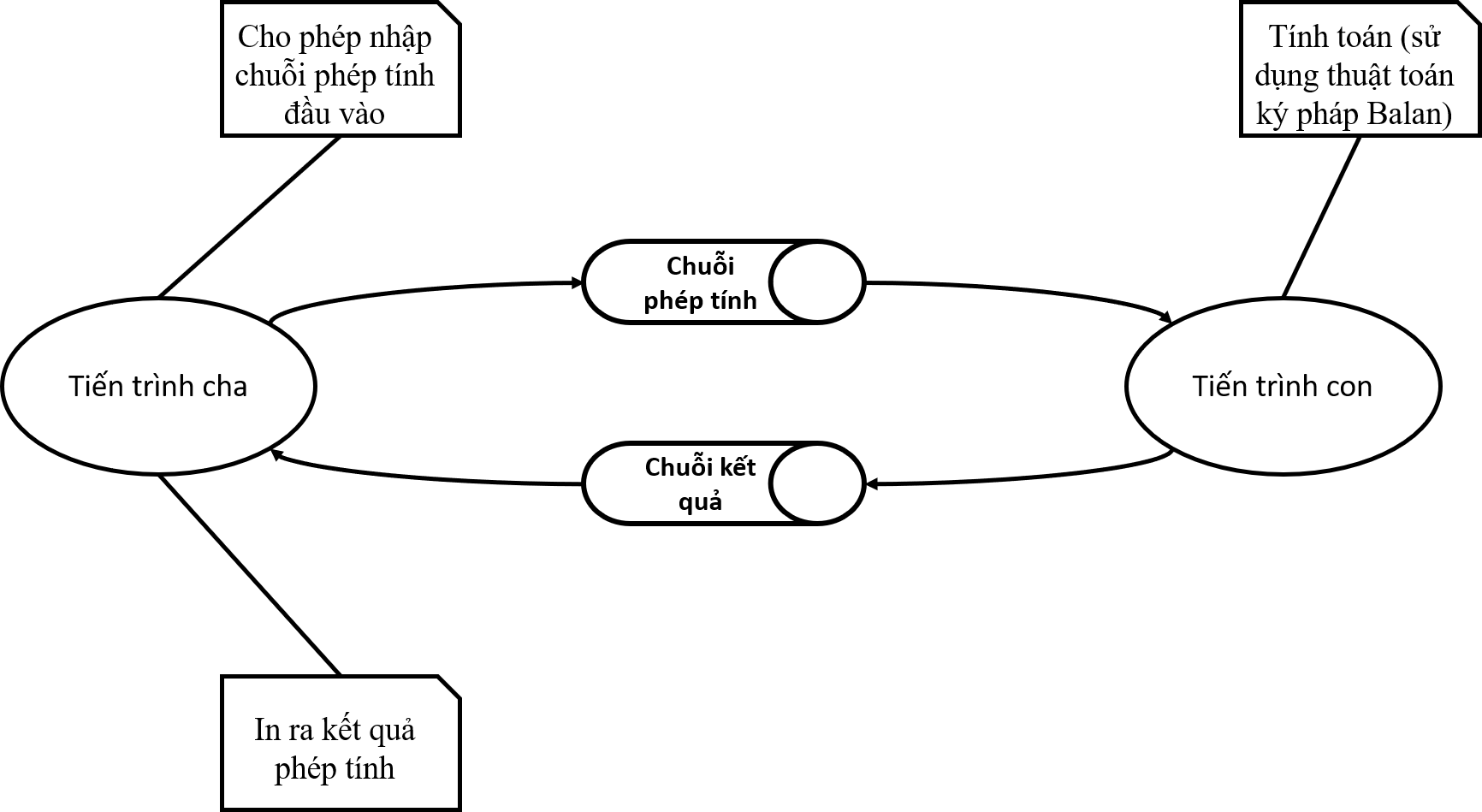
**Đầu ra:** Kết quả của phép tính.

*Ví dụ:* Với chuỗi “1+2\*sin(90)\*2 -1/4” thì ta sẽ có kết quả như sau:

**Đầu vào:** “1+2\*sin(90)\*2 -1/4”

**Đầu ra:** 4.75

- Từ việc phân tích bài toán ta tổ chức chương trình theo sơ đồ như sau:



1. Sơ đồ mô phỏng bài toán giao tiếp hai chiều giữa tiến trình cha-con bằng cơ chế đường ống.
2. **Thuật toán**

- Ta áp dụng chuyển biểu thức ở dạng trung tố (*infix*) sang dạng hậu tố (*posfix*) sử dụng thuật toán Ký pháp Balan.

1. Ví dụ về trung tố và hậu tố.

|  |  |
| --- | --- |
| **Infix** | **Posfix** |
| *1+2\*3+sin(45)\*cos(0)* | *1 2 3 \* + 45 sin 0 cos \* +* |
| *12+6/3\*cos(90)+1.5* | *12 6 3 / 90 cos \* + 1.5 +* |
| *3\*acos(1)-tan(0)/(2/6)* | *3 1 acos \* 0 tan 2 6 / / -* |
| *(1\*3)/((2+4)/8)* | *1 3 \* 2 4 + 8 / /* |

1. Mức độ ưu tiên của toán tử.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Toán tử** | *(* | *)* | *+* | *-* | *\** | */* | *%* | *sin* | *cos* | *tan* | *asin* | *acos* | *atan* | *^* | *sqrt* |
| **Độ ưu tiên** | *0* | *0* | *1* | *1* | *2* | *2* | *2* | *3* | *3* | *3* | *3* | *3* | *3* | *3* | *3* |

**Chuyển biểu thức dạng trung tố sang dạng hậu tố:**

**Bước 1:**

**-** Khởi tạo một stack rỗng, khởi tạo chuỗi kết quả (biến **result**) chính là chuỗi hậu tố của biểu thức.

**Bước 2:**

**-** Duyệt chuỗi phép tính từ trái sang phải rồi lấy ra từng toán hạng hoặc toán tử.

Vd: 10, 1, sin, cos, +, -, ), ( đặt biến chuỗi này là **str**.

- Kiểm tra str:

* Nếu **str** là dấu đóng ngoặc “(” thì push **str** vào stack.
* Nếu **str** là toán hạng thì nối vào chuỗi kết quả **result.**
* Nếu **str** là dấu đóng ngoặc “)” thì pop các phần tử trong stack và nối phần tử đó vào biến **result** cho đến khi gặp dấu đóng ngoặc “)” thì dừng (*lưu ý*: vẫn pop dấu đóng ngoặc nhưng không nối vào biến **result**).
* Nếu str là một toán tử:
* Kiểm tra nếu stack rỗng thì push str là stack.
* Kiểm tra nếu stack không rỗng, ta kiểm tra mức độ ưu tiên của toán tử **str** với toán tử là phần tử đầu tiên **top** của stack. Nếu mức ưu tiên của **str** <= mức ưu tiên của **top** thì ta pop phần tử **top** của stack và nối vào **result**, tiến hành lặp lại bước so sánh, ngược lại push **str** vào stack.

**Tính toán kết quả:**

**Bước 1:**

**-** Khởi tạo mảng kiểu float **data**[100];

**-** Biến **idx** = 0;

**Bước 2:**

**-** Duyệt chuỗi hậu tố từ trái sang phải rồi lấy ra từng toán hạng hoặc toán tử.

Vd: 10, 1, sin, cos, +, -, ), ( đặt biến chuỗi này là **str**.

- Kiểm tra str:

* Nếu **str** là toán hạng thì **data**[**idx**++] = **str**
* Nếu **str** là toán tử
* Nếu str là toán tử một toán hạng (*vd*: **str** = sin, cos, tan):

**data**[**idx** - 1] = **str**(data[idx - 1])

* Nếu str là toán tử hai toán hạng (*vd*: **str** = +, -, \*, /):

**data**[**idx** - 2] = **data**[**idx** - 2] **str** data[idx – 1)

- Duyệt cho đến khi kết thúc chuỗi, kết quả chương trình sẽ là **data**[0]

## CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ

1. **Biên dịch và thực thi**

- Hệ điều hành Linux (Ubuntu).

- Ngôn ngữ lập trình: C.

- Công cụ: Visual Studio Code.

- Các bước biên dịch và thực thi chương trình

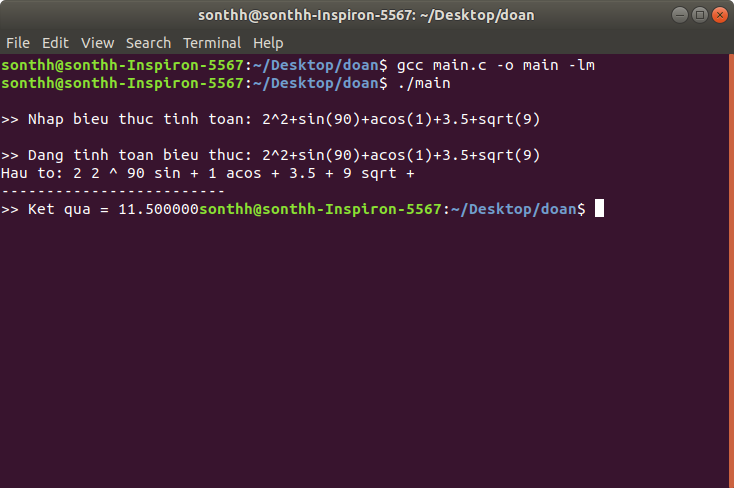
* Open terminal với thư mục đang chứa file mã nguồn main.c
* Chạy lệnh sau để biên dịch mã nguồn với trình biên dịch gcc

*gcc main.c –o main -lm*

* Chạy lệnh sau để thực thi chương trình sau khi đã được biên dịch

*./main*

* Nhập vào biểu thức và kiểm trả kết quả



1. Kết quả demo chương trình
2. **Đánh giá**

Ưu điểm:

* Hoạt động tốt với những phép toán không quá phức tạp, kết quả nhanh không bị delay.
* 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, học cách sử dụng hệ điều hành Linux, các lệnh cơ bản và thao tác trong Terminal của Linux.
* Áp dụng để xây dựng chương trình giải quyết bài toán theo yêu cầu đặt ra.

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.
* Chương trình mô phỏng còn đơn giản, không đảm bảo được tính đúng đắn với các phép tính toán phức tạp.

## CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận**

Bài toán trên vẫn còn 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 ảnh 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.

+ Cải thiện được kĩ năng lập trình với ngôn ngữ C và kĩ năng sử dụng hệ điều hành Linux, thao tác với Teminal trong Linux.

* 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 đảm bảo được các ngoại lệ có thể xảy ra trong quá trình tính toán các phép tính toán phức tạp.

1. **Hướng phát triển**

* Tiếp tục nghiên cứu lý thuyết và thực hành về cơ chế giao tiếp giữa các tiến trình, sử dụng ngôn ngữ lập trình C và hệ điều hành Linux.
* Khắc phục được những ngoại lệ xảy ra trong tính toán.
* Xây dựng thuật toán hoàn hiện hơn với các phép tính phức tạp.
* Xây dựng với giao diện người dùng.

# PHẦN 2: LẬP TRÌNH MẠNG

**ĐỀ TÀI: TÌM HIỂU GIAO THỨC SMTP, POP3**

**XÂY DỰNG CHƯƠNG TRÌNH MAIL-CLIENT**

**Yêu cầu đề tài**

* Tìm hiểu các giao thức gởi/nhận mail.
* Tìm hiểu bộ thư viện Javamail API.
* Tìm hiểu mô hình lập trình MVC với JSP, Servlet để xây dựng chương trình demo.

## CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. **Các giao thức gửi/nhận mail**

**Giao thức SMTP:**

- SMTP là viết tắt của Simple Mail Transfer Protocol hay giao thức truyền tải thư tín đơn giản, là một chuẩn truyền tải thư điện tử qua mạng Internet.

- SMTP được dùng để liên lạc với server từ xa và gửi email từ mail client tới mail server và sau đó mail được gửi đến mail server của người nhận. Quá trình này được điều khiển bởi Mail Transfer Agent (**MTA**) trên email server của bạn. Cũng vậy, SMTP chỉ được dùng cho mục đích gửi email.

- Dưới đây là danh sách port của các smtp mail server thông dụng

1. Danh sách port của các smtp mail server thông dụng

|  |  |  |
| --- | --- | --- |
|  | Server | Port |
| Google gmail | smtp.gmail.com | 465 |
| smtp.gmail.com | 587 |
| Outlook.com | smtp.gmail.com | 587 |
| Yahoo mail | smtp.mail.yahoo.com | 465 |
| Yahoo UK | smtp.mail.yahoo.co.uk. | 465 |

**Giao thức POP3:**

- Post Office Protocol Version 3 (Pop3) là một giao thức chuẩn trên Internet cho phép một workstation có thể truy xuất động đến một maildrop trên một server từ xa. Có nghĩa là Pop3 được dùng để cho phép workstation lấy mail mà server đang giữ nó.

- Port chuẩn dành cho dịch vụ Pop3 được qui ước là TCP port 110. Pop3 server sẽ khởi động và lắng nghe trên port này. Một client muốn sử dụng các dịch vụ của Pop3 thì nó phải thiết lập một kết nối tới Pop3 server. Khi kết nối được thiết lập thì Pop3 server sẽ gởi tới client một lời chào. Sau đó, Pop3 Client và Pop3 Server sau đó trao đổi các request và reply cho đến khi kết nối được đóng hay loại bỏ.

- Một Pop3 session sẽ phải trải qua các trạng thái: xác nhận (Authorization), giao dịch (transaction) và trạng thái cập nhật (Update).

- Trong trạng thái xác nhận, client phải thông báo cho server biết nó là ai. Khi server đã xác nhận được client, session sẽ đi vào trạng thái giao dịch. Trong trạng thái này, client hoạt động bằng cách gởi các request tới server. Khi client gởi lệnh “QUIT”, session sẽ đi vào trạng thái cập nhật (Update). Trong trạng thái này, Pop3 server giải phóng các tài nguyên và gởi lời tạm biệt. Sau đó kết nối TCP đóng lại.

- Một Pop3 server có một khoảng thời gian time out. Khi xảy ra time out, session không đi vào trạng thái cập nhật (Update) mà server sẽ tự đóng kết nối TCP mà không xoá bất kỳ message nào hay gởi đáp ứng cho client.

**Giao thức INAP:**

- Internet Message Access Protocol (IMAP) cung cấp lệnh để phần mềm thư điện tử trên máy khách và máy chủ dùng trong trao đổi thông tin phiên bản 4 (IMAP4rev1). Đó là phương pháp để người dùng cuối truy cập thông điệp thư điện tử hay bản tin điện tử từ máy chủ về. Nó cho phép chương trình thư điện tử dùng cho máy khách như Netscape Mail, Eudora của Qualcomm, Lotus Notes hay Microsoft Outlook lấy thông điệp từ xa trên máy chủ một cách dễ dàng như trên đĩa cứng cục bộ.

- IMAP khác với giao thức truy cập thư điện tử Post Office Protocol (POP). POP lưu trữ toàn bộ thông điệp trên máy chủ. Người dùng kết nối bằng đường điện thoại vào máy chủ và POP sẽ đưa các thông điệp vào inbox của người dùng. Hai giao thức này đã được dùng từ hơn 10 năm nay. Theo một nhà phân tích thì khác biệt chính giữa POP (phiên bản hiện hành 3.0) và IMAP (phiên bản hiện hành 4.0) là POP3 cho người dùng ít quyền điều khiển hơn trên thông điệp.

- IMAP là cơ chế cho phép lấy thông tin về thư điện tử của bạn, hay chính các thông điệp từ mail server của môi trường cộng tác. Giao thức thư điện tử này cho phép người dùng kết nối vào máy chủ Internet từ xa, xem xét phần tiêu đề và người gửi của thư điện tử trước khi tải những thư này về máy chủ của mình. Với IMAP người dùng có thể truy cập các thông điệp như chúng được lưu trữ cục bộ trong khi thực tế lại là thao tác trên máy chủ cách xa hàng kilômet. Với khả năng truy cập từ xa này, IMAP dễ được người dùng cộng tác chấp nhận vì họ coi trọng khả năng làm việc lưu động.

- Một kết nối của IMAP4rev1 được thành lập theo một kết nối Client/Server và sự tương tác trao đổi thông tin hay lấy mail về từ Server của người sử dụng thông qua các lệnh truy suất mà IMAP4rev1 đã định dạng sẵn trong giao thức IMAP.

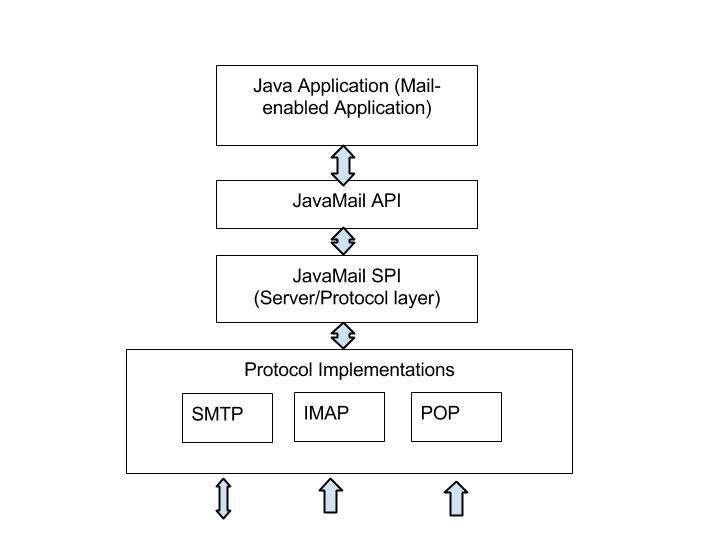
1. **JavaMail API**

- JavaMail API cung cấp một bộ thư viện độc lập giao thức và độc lập nền tảng để xây dựng các chương trình gửi/nhận thư điện tử. Nó cung cấp một bộ package bao gồm các abstract class phục vụ cho việc đọc, soạn và gởi thư/tin nhắn điện tử.

- JavaMail API bao gồm các class mà thực hiện các tiêu chuẩn tin nhắn Internet RFC822 và MIME.

- Một số giao thức được hỗ trợ bởi Javamail API: SMTP, POP3, INAP, INET, MIME, NNTP

- Dưới đây là hình vẽ minh họa kiến trúc của JavaMail API



1. Kiến trúc của JavaMail API

- JavaMail được chia thành hai phần chính:

* Phần độc lập ứng dụng: API được sử dụng bởi các ứng dụng để gửi và nhận thư, độc lập với nhà cung cấp hoặc các giao thức mà được sử dụng.
* Phần phụ thuộc dịch vụ: như các giao thức SMTP, POP3, INET, MIME, NNTP, phụ thuộc và nhà cung cấp dịch vụ mail.

- Trang chủ của JavaMail API: *https://javaee.github.io/javamail*

## CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ

1. **Phân tích chức năng**

Hệ thống bao gồm các chức năng sau:

**Đăng nhập:** Cho phép người dùng sử dụng tài khoản email của mình để đăng nhập vào để sử dụng chương trình mail client.

**Đăng xuất:** thoát khỏi hệ thống, hủy phiên làm việc hiện tại.

**Xem danh sách các email:** sau khi đăng nhập thành công, người dùng có thể xem email của mình đã được phân loại theo thư mục (người dùng có thể lựa chọn bên thanh menu bên trái)

* **Hộp thư đến:** Liệt kê các email đã được nhận
* **Thư đã gửi:** Liệt kê các email đã gửi
* **Thư nháp:** Liệt kê các email nháp
* **Thư spam:** Liệt kê các email là spam

**Xem chi tiết email:** Xem chi tiết toàn bộ thông tin email bao gồm: tiêu đề, người gửi, danh dánh người nhận (To), danh sách BCC, CC, ngày gửi, nội dung email (dạng HTML nếu có), hiển thị danh sách file đính kèm (hình ảnh hoặc tài liệu).

**Download file đính kèm:** Khi xem chi tiết email người dùng có thể download file đính kèm (nếu có).

**Soạn và gởi email:** chương trình cho phép người dùng soạn và gởi một email với đầy đủ các thông tin: tiêu đề, danh dánh người nhận (To), danh sách BCC, CC, ngày gửi, nội dung email, đính kèm hình ảnh hoặc file tài liệu.

**Xóa email:** Cho phép người dùng có thể xóa email, có thể xóa nhiều email cùng một lúc.

1. **Thiết kế**

**Cấu trúc chương trình:** được thiết kế dựa trên mô hình MVC (sử dụng JSP, Servlet) như sau:



1. Tổ chức project theo mô hình mvc

**Controller:** là nới tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class, phương thức xử lý điều hướng, giúp lấy đúng dữ liệu thông tin cần thiết nhờ các lớp Model (Entity và Service) cung cấp và hiển thị dữ liệu đó ra cho người dùng gói View.

* LoginController: Servlet điều hướng yêu cầu sử dụng dụng hệ thống của người dùng, nếu đăng nhập đúng tài khoản email, thì được vào sử dụng hệ thống, ngược lại hiển thị thông báo lỗi.
* LogoutController: Servlet hủy phiên đăng nhập của người dùng.
* InboxController: Servlet thực hiện chức năng controller cho email thuộc folder INBOX.
* SentController: Servlet thực hiện chức năng controller cho email thuộc folder [Gmail]/Sent Mail
* DraftController: Servlet thực hiện chức năng controller cho email thuộc folder [Gmail]/Drafts.
* SpamController: Servlet thực hiện chức năng controller cho email thuộc folder [Gmail]/Spam.
* ReadController: Servlet thực hiện chức năng controller thực hiện việc đọc thông tin của một email cụ thể.
* DownloadController: Servlet thực hiện chức năng donwload file đính kèm trong email (nếu có).
* DeleteController: Servlet thực hiện chức năng xóa một hoặc nhiều email cùng một lúc.
* ComposeController: Servlet thực hiện chức năng controller cho việc soạn và gửi email.

**View:** là nới chứa những giao diện như một nút bấm, ô nhập liệu, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu từ controller giúp người dùng tương tác với hệ thống.

Bao gồm các file \*.jsp (view) tương ứng với một Servlet (controller)

* login.jsp: chứa giao diện trang /login.
* inbox.jsp: chứa giao diện trang /inbox.
* sent.jsp: chứa giao diện trang /sent.
* draft.jsp: chứa giao diện trang /draft.
* spam.jsp: chứa giao diện trang /spam.
* read.jsp: chứa giao diện trang /read.
* compose.jsp: chứa giao diện trang /compose.

**Model:** là nơi chứa những nghiệp vụ tương tác với dữ liệu email, nó sẽ bao gồm các class, phương thức xử lý nhiều nghiệp vụ như lấy dữ liệu email, xóa email, gởi email… Trong đó gói **Entity** bao gồm tất cả class thực thể như Gmail, Attachment, UserLogin…, còn **Service** chứa các class xử lí nghiệp vụ về xử lí email để phục vụ cho gói **Controller**.

* Gmail: lớp chưa thông tin của một email.
* messageNumber: id của một tin.
* subject: tiêu đề gmail.
* content: nội dung gmail.
* from: người gởi.
* to, cc, bcc.
* date: ngày gửi.
* attachments: danh sách file đính kềm.
* Atachment: lớp lưu trữ thông tin của một file đính kèm.
* PageGmail: lớp hỗ trợ xử lý phân trang vì có nhiều email.
* UserLogin: lớp chứa thông tin tài khoản người dùng.
* FetchingEmailService: lớp nghiệp vụ xử lý liên quan đến việc kết nối mail server để get/delete email.
* SendingEmailService: lớp nghiệp vụ xử lý liên quan đến việc kết nối mail server để gửi một mail.

## CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ

1. **Môi trường phát triển**

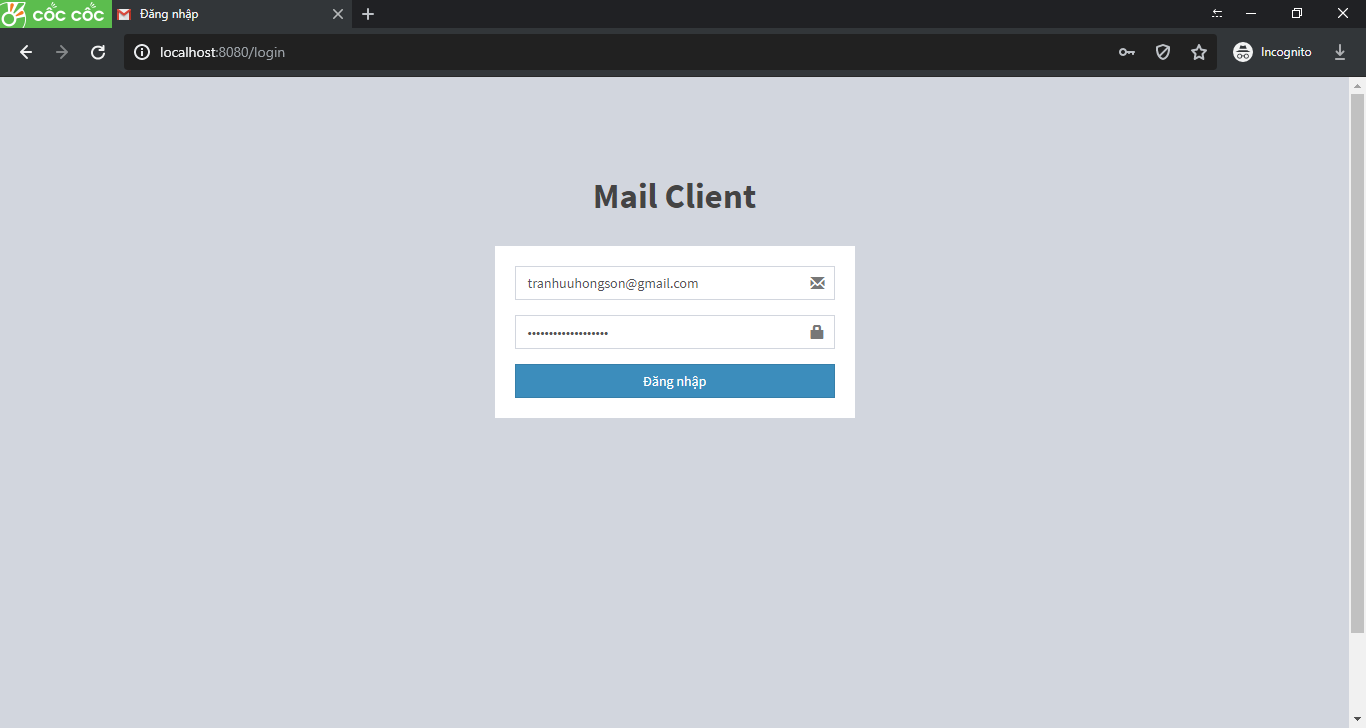
- Ngôn ngữ: Java (Servlet, JSP), HTML, CSS

- Môi trường: JDK 1.8

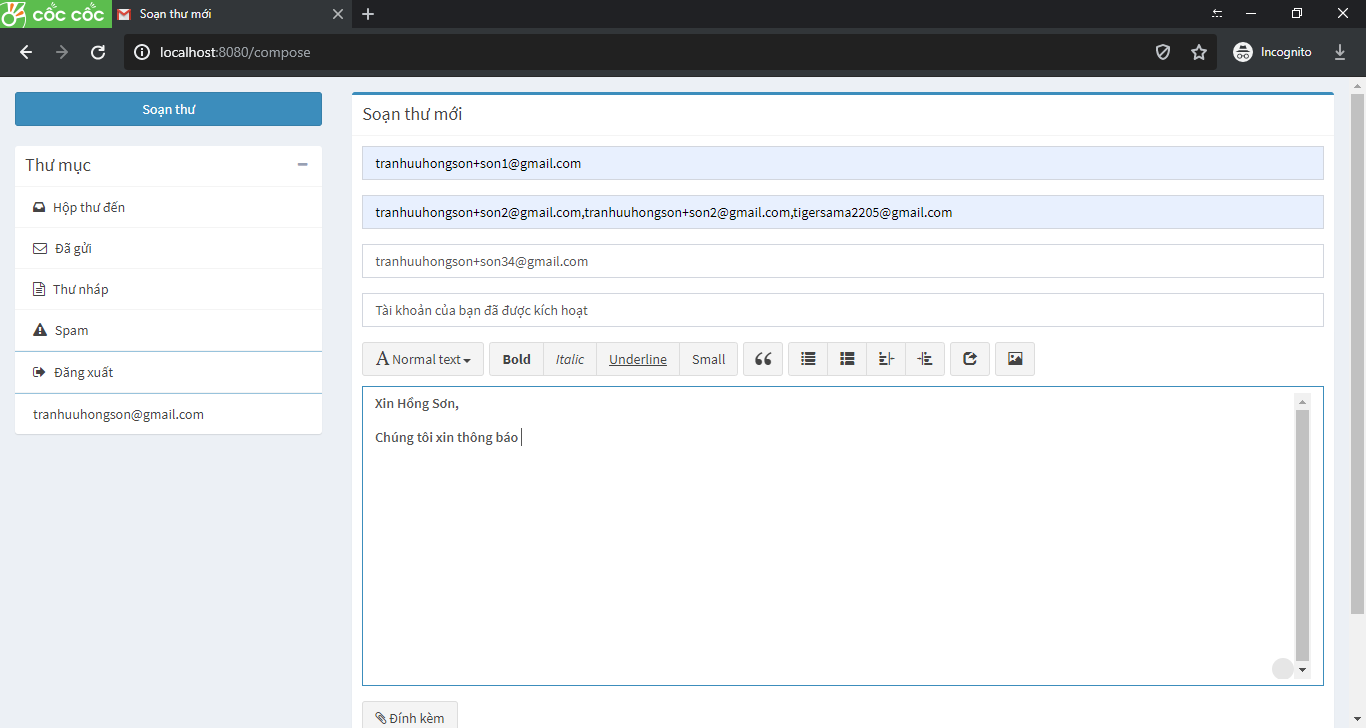
- Webserver: Apache Tomcat 8.1

- Công cụ: Eclipse IDE

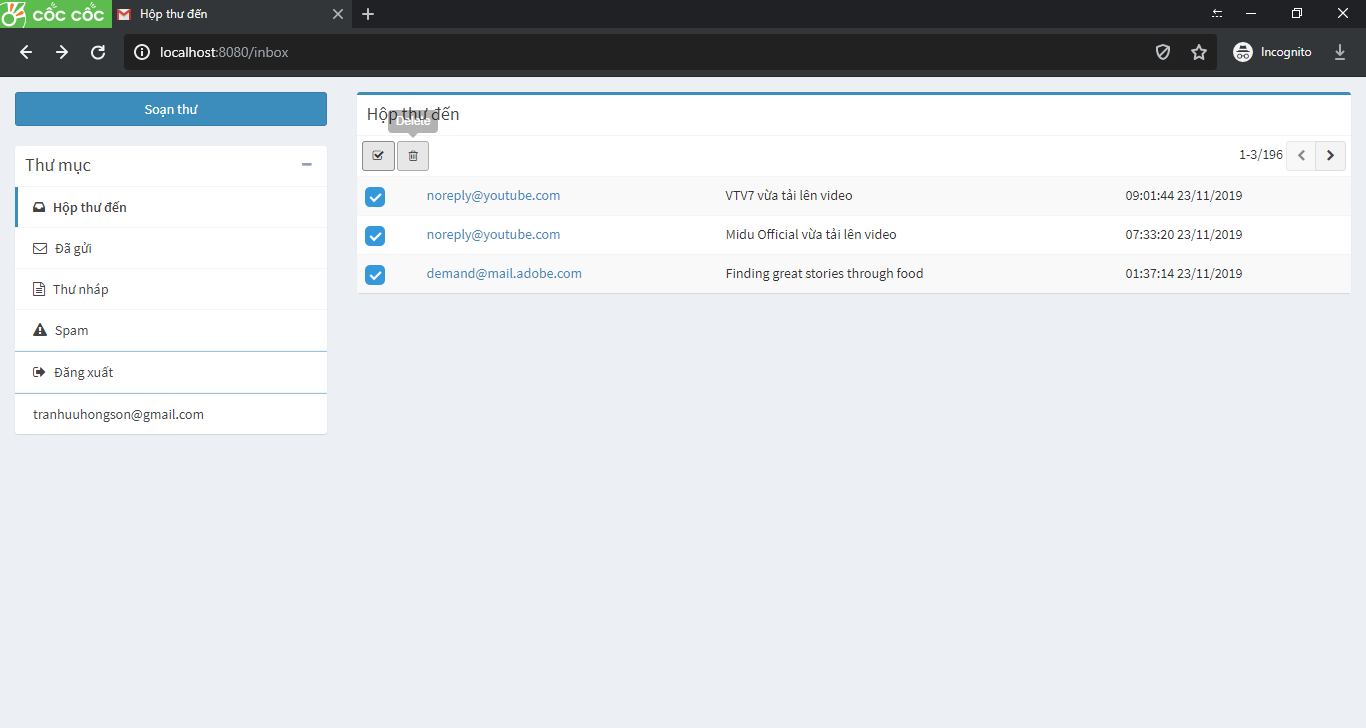
1. **Demo chương trình**



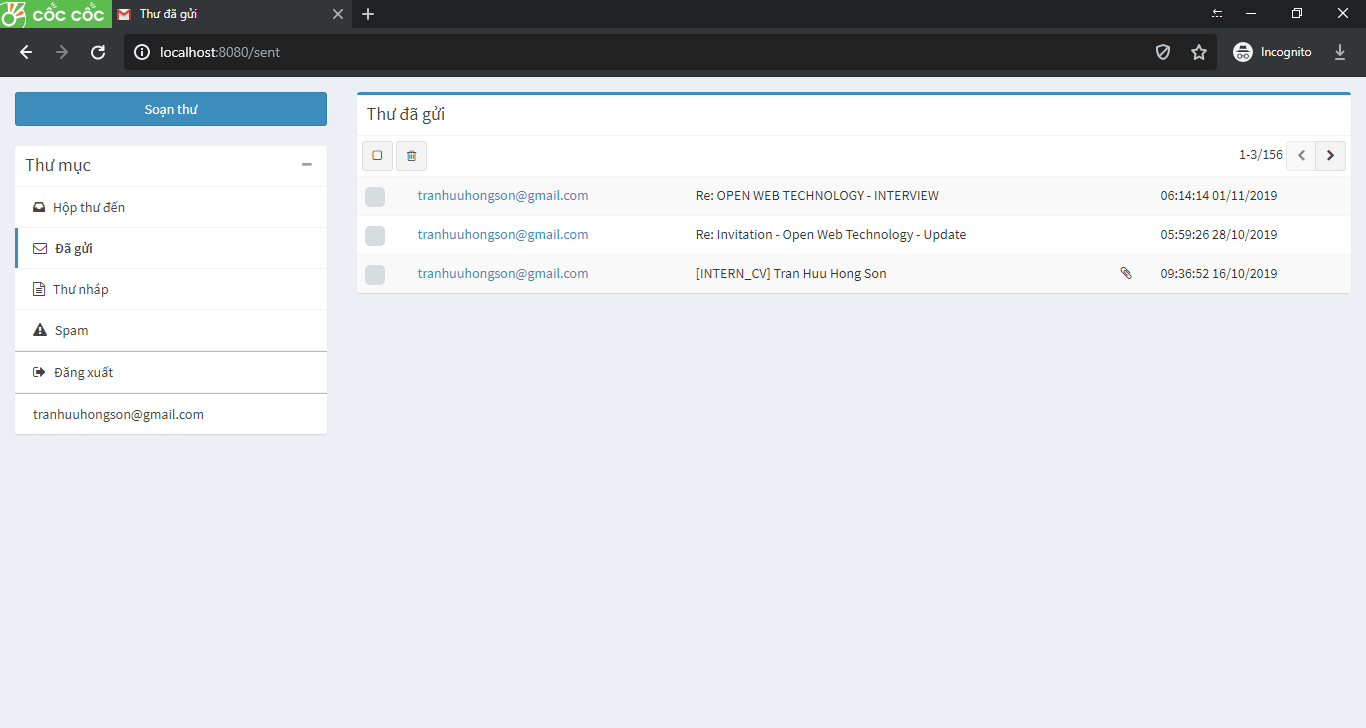
1. Màn hình trang đăng nhập



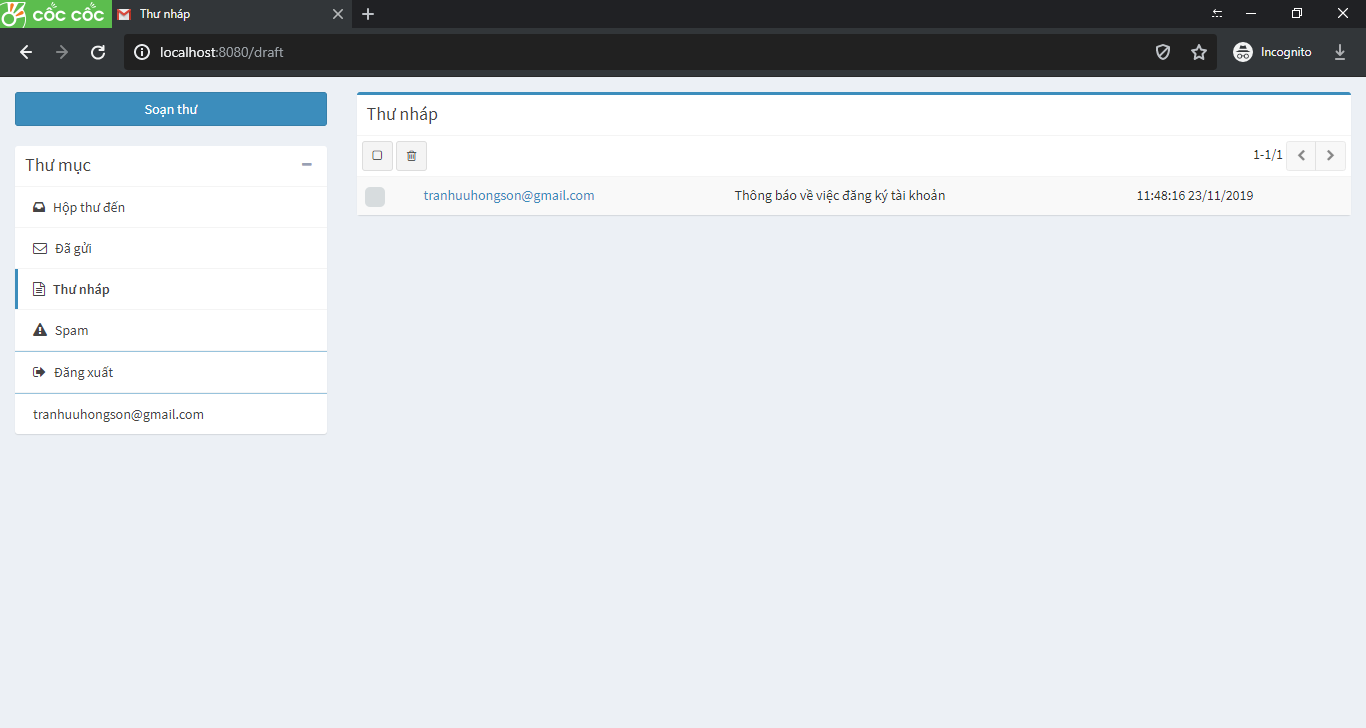
1. Màn hình trang soạn và gởi email



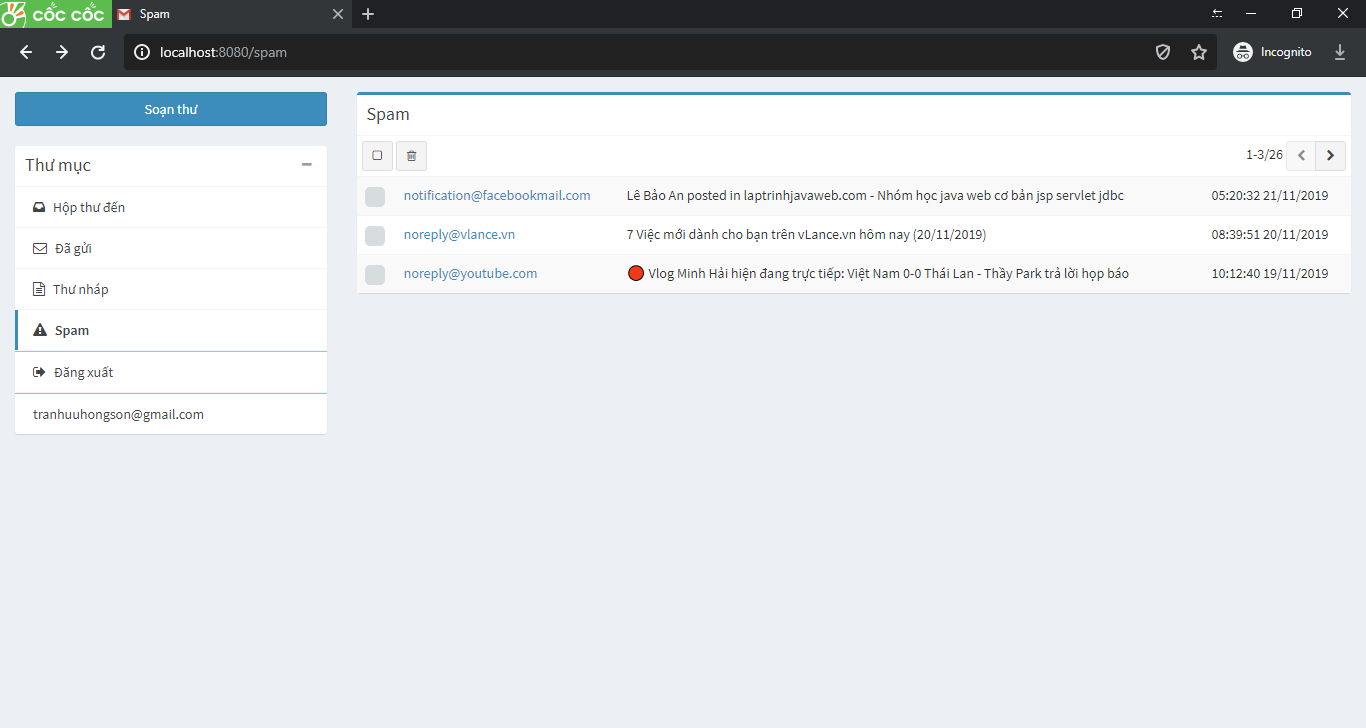
1. Màn hình trang hiển thị email thư mục INBOX



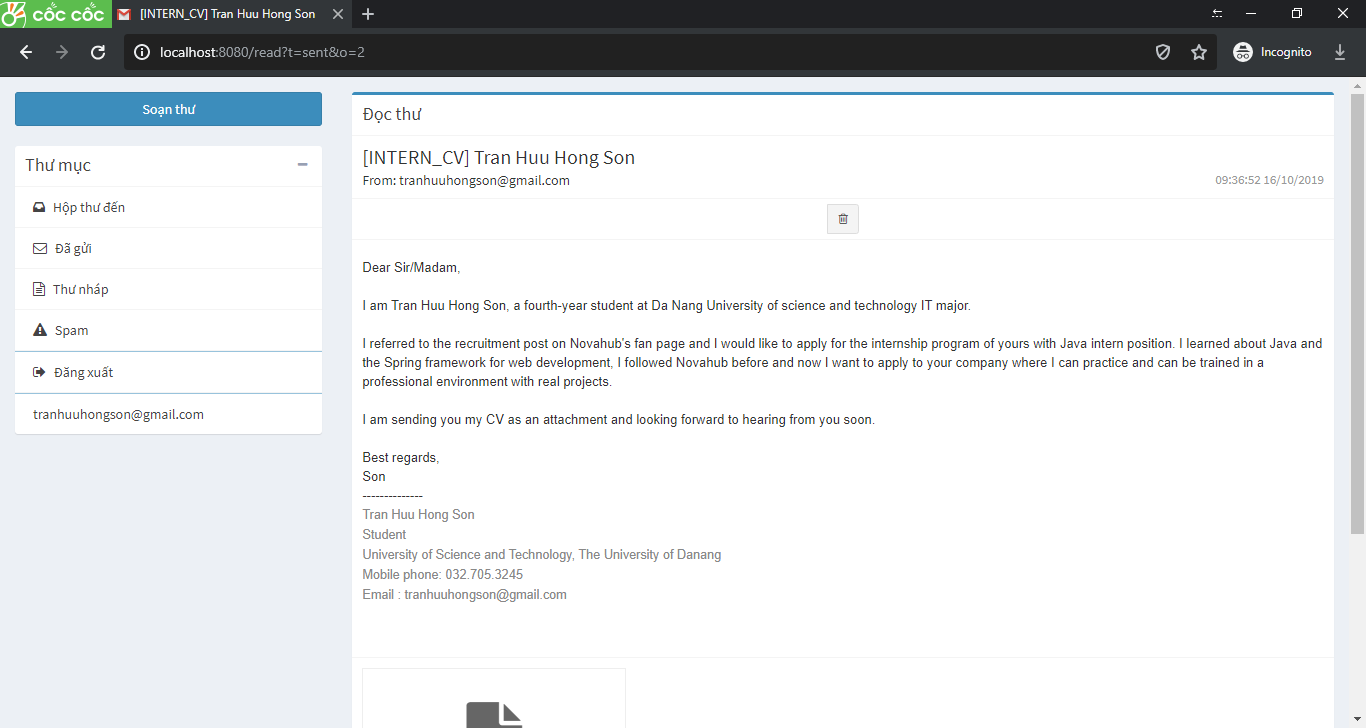
1. Màn hình trang hiển thị email thư mục [Gmail]/Sent Mail



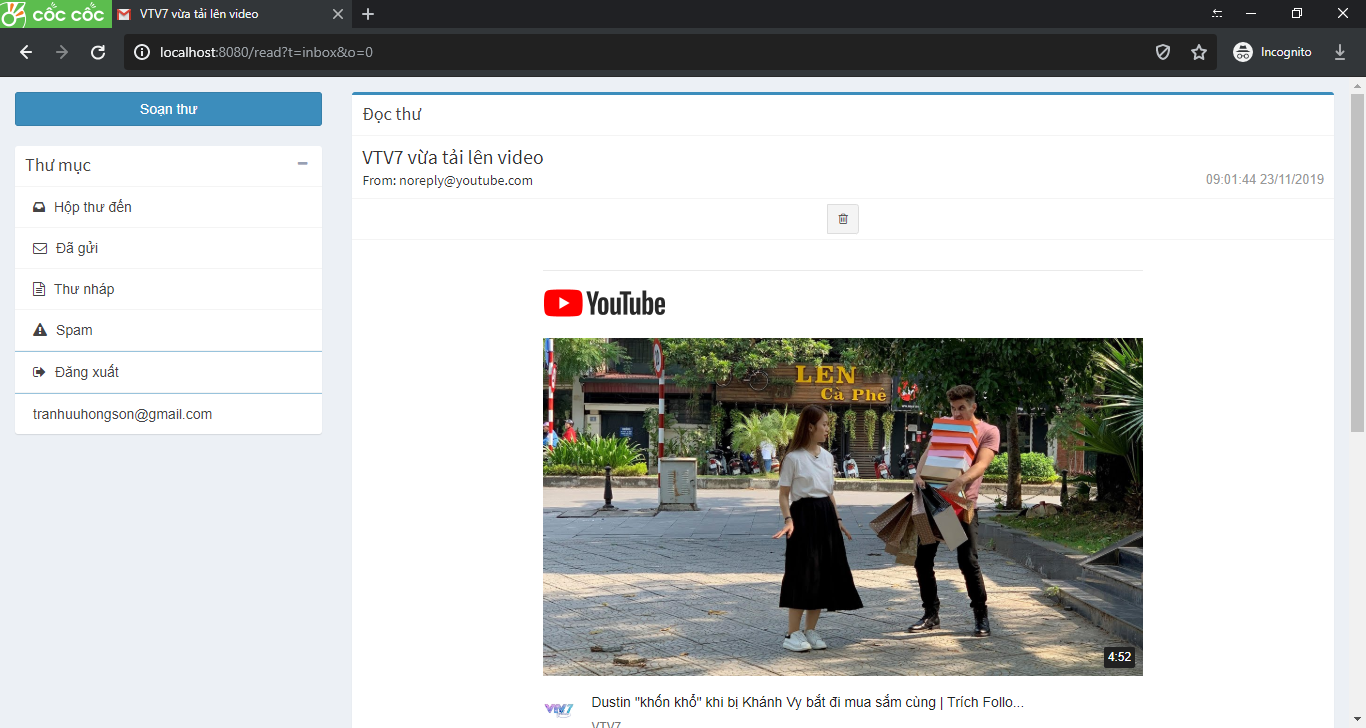
1. Màn hình trang hiển thị email thư mục [Gmail]/Drafts



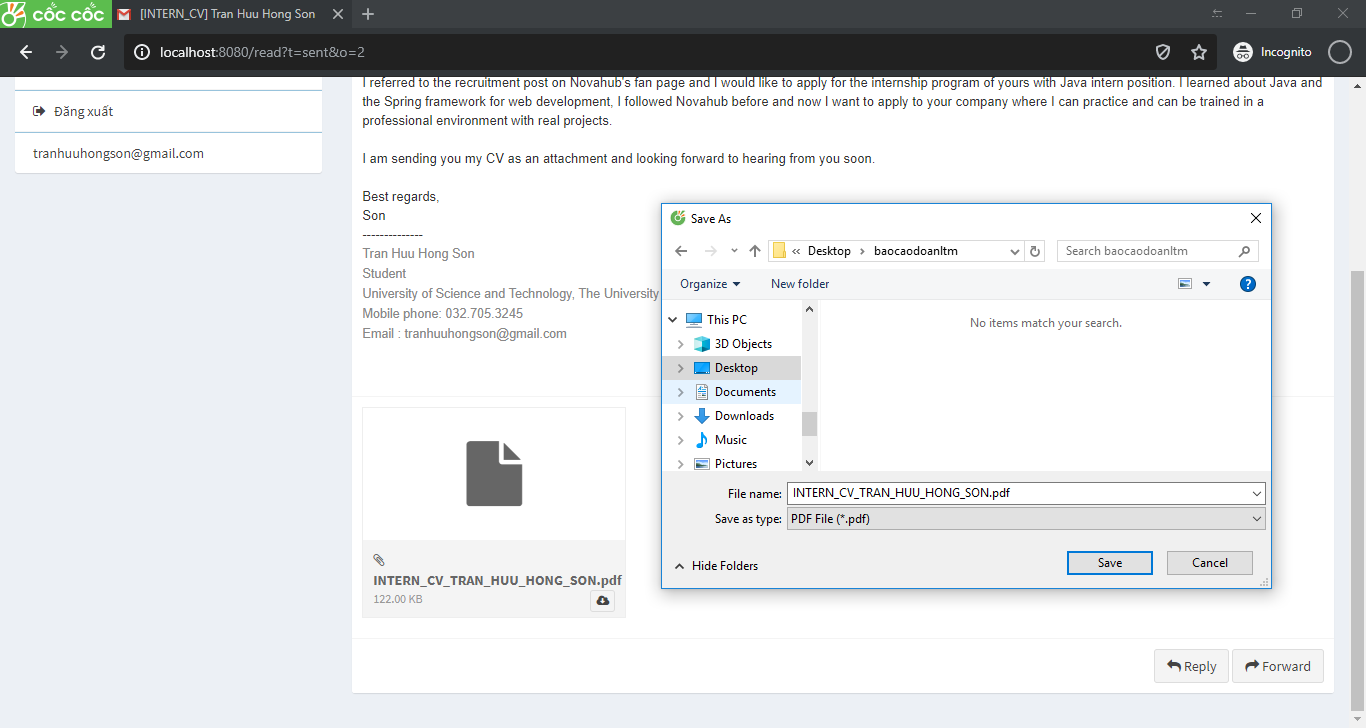
1. Màn hình trang hiển thị email thư mục [Gmail]/Spam



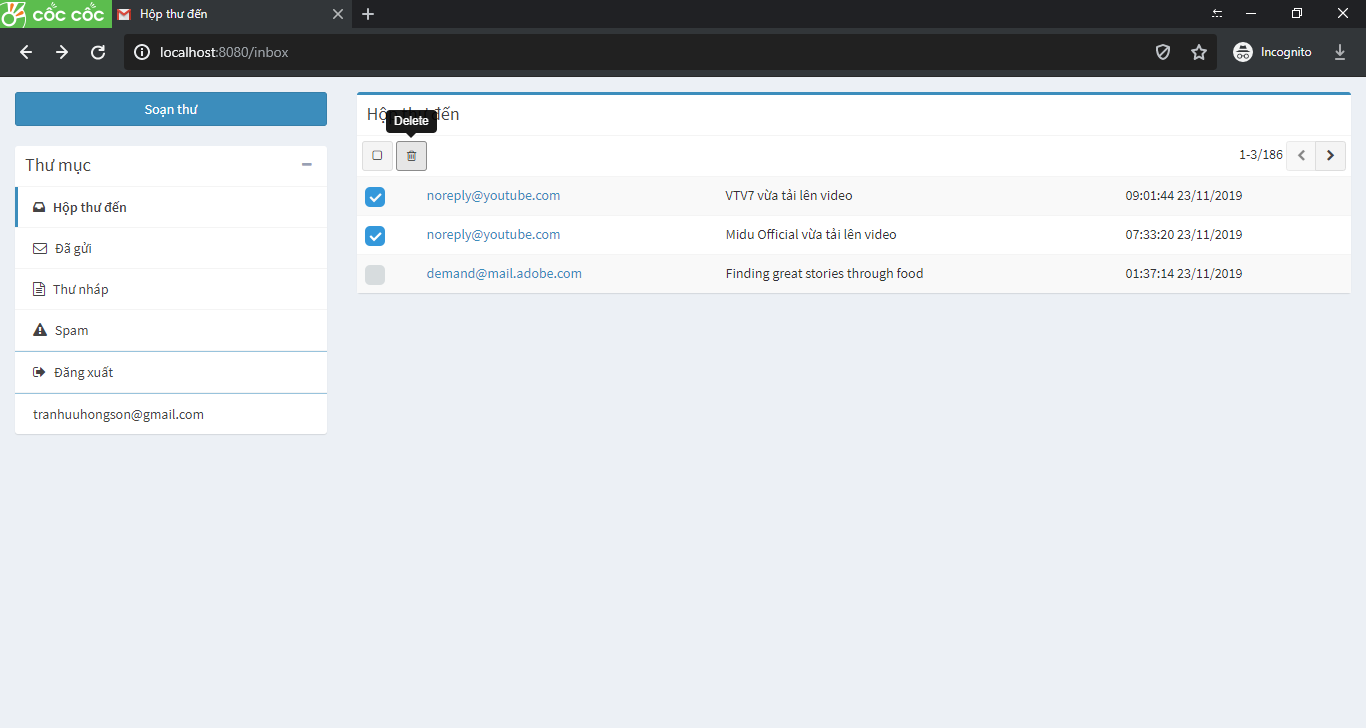
1. Màn hình hiển thị trang xem chi tiết email



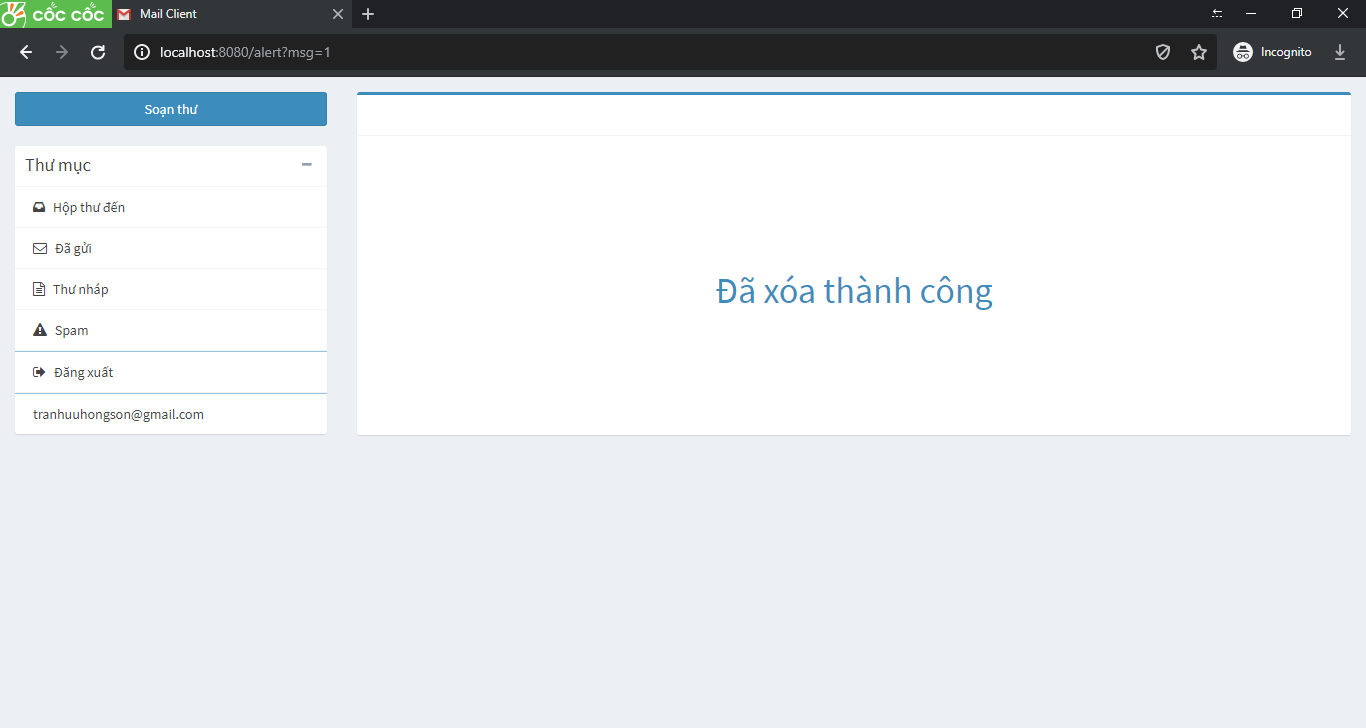
1. Màn hình hiển thị trang xem chi tiết email có nội dung HTML



1. Màn hình hiển thị download file đính kèm



1. Màn hình hiển thị việc xóa email



1. Màn hình hiển thị việc xóa email thành công
2. **Đánh giá:**

Ưu điểm:

* Hoạt động ổn định.
* Đảm bảo tính đúng đắn của dữ liệu email.
* Có giao diện dễ sử dụng.
* Đáp ứng các chức năng cơ bản của một chương trình mail-client như: gửi email, đọc email (nhiều loại thư mục), xóa email, đính kèm tệp tin, download tệp tin, hiển thị hình ảnh, email dạng HTML.

Nhược điểm:

* Lấy dữ liệu từ mail server còn chậm.
* Thiếu một số chức năng như các chương trình mail client thật mà ta vẫn hay dùng như: phân loại thư mục thư như Promotion, Social.
* Chưa có chức năng dánh dấu sao cho thư, đánh đấu đã đọc và đánh dấu spam.

## CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận**

Trong thời gian tìm hiểu và nghiên cứu cơ sở lý thuyết và triển khai lập trình, việc thực hiện đồ án đã đạt được những kết quả như sau:

- Về mặt lý thuyết, đồ án đã đạt được các nội dung cơ bản liên quan đến các giao thức gửi nhận mail như SMTP, POP3, IMAP, cấu trúc một thư điện tử email.

- Về mặt thực tiễn ứng dụng, đồ án đã đạt được việc mô phỏng chương trình mail client gồm các chức năng gửi ,nhận email, đọc, xóa email.

Tuy nhiên, đồ án còn tồn tại các vấn đề như sau:

- Còn thiếu một số tính năng so với hệ thống gmail thực tế mà người ta hay sử dụng.

- Xây dựng giao diện trực quan hơn, gần gũi giống như những chương trình mail-client hay dùng.

1. **Hướng phát triển**

Một số hướng nghiên cứu và phát triển của đề tài như sau:

- Thêm các hộp thư khác như hộp thư quan trọng (important), thùng rác (trash), các thư mục Social, Promotion.

- Thêm các chức năng như mark started, mark as read.

- Xây dựng giao diện trực quan hơn.

**TÀI LIỆU THAM KHẢO**

1. ThS. Trần Hồ Thủy Tiên, ĐH Bách Khoa Đà Nẵng, Slide Nguyên lý hệ điều hành
2. Nguyễn Phương Lan, Hồ Đức Hải, Lập trình Linux Tập 1
3. ThS. Mai Văn Hà, ĐH Bách Khoa Đà Nẵng, Slide Lập trình mạng
4. JavaMail API Tutorial: [https://www.tutorialspoint.com/javamailapi/index.htm](https://www.tutorialspoint.com/javamail_api/index.htm)