ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
CƠ SỞ NGÀNH MẠNG

Đề tài:

Phần hệ điều hành:

Xây dựng ứng dụng truyền thông điệp bằng Pipe và Shared Memory để tính biểu thức toán học

Phần lập trình mạng:

Xây dựng ứng dụng ngân hàng theo mô hình Client-Server sử dụng cơ sở dữ liệu tập trung

GVHD : **Ths Mai Văn Hà**

SVTH : **NGUYỄN VĂN ĐẠI**

MSSV : **102170076**

LỚP : **17T2**

Đà Nẵng, 12/2020

Mục lục

PHẦN I: HỆ ĐIỀU HÀNH

Đề tài: Dùng pipe và shared memory để giao tiếp giữa hai quá trình. Quá trình thứ nhất cho người dùng nhập vào từ bàn phím một chuỗi biểu diễn một phép tính gồm các phần tử +, -, *, /, ^, sqrt, sin, cos, tan.

Ví dụ: 2+12+(12-4-6) *((3+4)-5)

(11+(-2)–((3/4)-5))

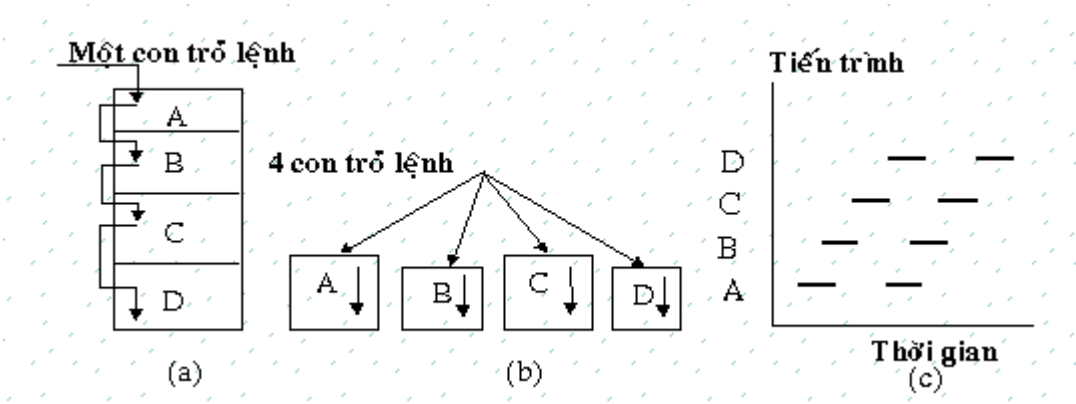
Truyền chuỗi dữ liệu này cho quá trình thứ hai. Quá trình thứ hai thực hiện tính

toán trên và trả kết quả về cho quá trình thứ nhất để hiển thị cho người sử dụng biết.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

I. Bối cảnh

Để hỗ trợ sự đa chương trình, máy tính yêu cầu phải chạy đa tác vụ đồng thời. Việc điều khiển nhiều hoạt động song song trên mặt vật lý phần cứng là rất khó khăn. Vì thế các nhà thiết kế đã đề xuất mô hình *song song giả lập* bằng cách thực thi qua lại giữa các chương trình để duy trì nhiều chương trình cùng một lúc, điều này tạo cảm giác cho người dùng như đang thực thi nhiều chương trình đồng thời.



Hình 1. Các chương trình xử lý song song giả lập

Tất cả các phần mềm, ứng dụng trong hệ thống được chia thành nhiều các tiến trình nhỏ. 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, mỗi tiến trình cần một số tài nguyên – như CPU, bộ nhớ chính, các tệp tin và thiết bị nhập xuất.

Khi các tiến trình thực thi, không tránh khỏi việc sử dụng chung tài nguyên hay trực tiếp truyền thông điệp cho nhau để thuận tiện cho việc xử lý tác vụ. Để giao tiếp giữa các tiến trình như vậy mà không cần xem xét đến phần giao tiếp mức thấp, ta cần tìm hiểu đến các cơ chế giao tiếp giữa các tiến trình (IPC).

II. Mục tiêu

Mục tiêu:

- Hiểu khái quát về tiến trình, ứng dụng
- Hiểu rõ về giao tiếp giữa các tiến trình và cách ứng dụng một số cơ chế giao tiếp giữa các tiến trình để truyền thông điệp (Cụ thể là Pipe và Shared Memory).
- Tìm hiểu về thuật toán Ký pháp Ba Lan và cài đặt ứng thuật toán giải được nhiều biểu thức toán học.
- Nâng cao kỹ năng lập trình Ngôn ngữ C/C++

III. Giới thiệu về giao tiếp giữa các tiến trình

Tiến trình là gì?

Một tiến trình trong hệ điều hành có thể là tiến trình độc lập hoặc là tiến trình hợp tác.

Tiến trình độc lập là tiến trình không bị ảnh hưởng bởi sự thực thi của các tiến trình khác. Tiến trình độc lập sẽ không chia sẻ bất kỳ dữ liệu nào với tiến trình khác. Trong khi tiến trình hợp tác thì ngược lại.

Tại sao các tiến trình phải hợp tác?

Chia sẻ thông tin: Nhiều người dùng cần truy cập chung một thông tin, nên cần có môi trường để truy cập đồng thời những loại tài nguyên này.

Tăng tốc độ tính toán: Để tác vụ xử lý nhanh hơn, thì tác vụ nên được chia nhỏ hơn, mỗi tác vụ con sẽ được thực thi song song trên các tiến trình. Tốc độ cải thiện nếu máy tính có nhiều thành phần xử lý như CPU hay kênh I/O.

Module hóa: Xây dựng hệ thống theo lối module hóa, bằng cách chia các chức năng của hệ thống thành các tiến trình, luồng nhỏ.

Giao tiếp giữa các tiến trình

Hình 2. Mô phỏng giao tiếp giữa các tiến trình

Giao tiếp giữa các tiến trình (IPC – Inter-Process Communication) hay còn gọi là giao tiếp liên tiến trình là **cơ chế thiết lập kết nối giữa các tiến trình** để thực hiện **truyền các dữ liệu hoặc thông điệp**. Nhìn chung, các ứng dụng dùng IPC được phân loại thành nhóm client hoặc nhóm server. Phía ứng dụng, tiến trình client gửi các yêu cầu dịch vụ đến các ứng dụng, tiến trình khác. Phía ứng dụng, tiến trình server sẽ hồi đáp lại các yêu cầu của các client. Nhiều ứng dụng cũng có thể hoạt động với vai trò vừa như 1 client vừa như 1 server, tùy vào tình huống sử dụng.

Mục đích sử dụng IPC: Sử dụng các cơ chế IPC khi cần giao tiếp giữa các chương trình, khi ta muốn giao tiếp nhanh hơn, không cần quan tâm đến giao tiếp mức thấp giữa các tiến trình.

Các cơ chế hỗ trợ trên nền tảng Linux:

- Pipe – Trao đổi bằng đường ống
- Shared Memory – Trao đổi bằng bộ nhớ chia sẻ

- Signals handling – Trao đổi bằng tín hiệu
- Message Queues – Trao đổi bằng Hàng đợi tin nhắn
- Giao tiếp thông qua Socket
- Giao tiếp đồng bộ dùng Semaphore

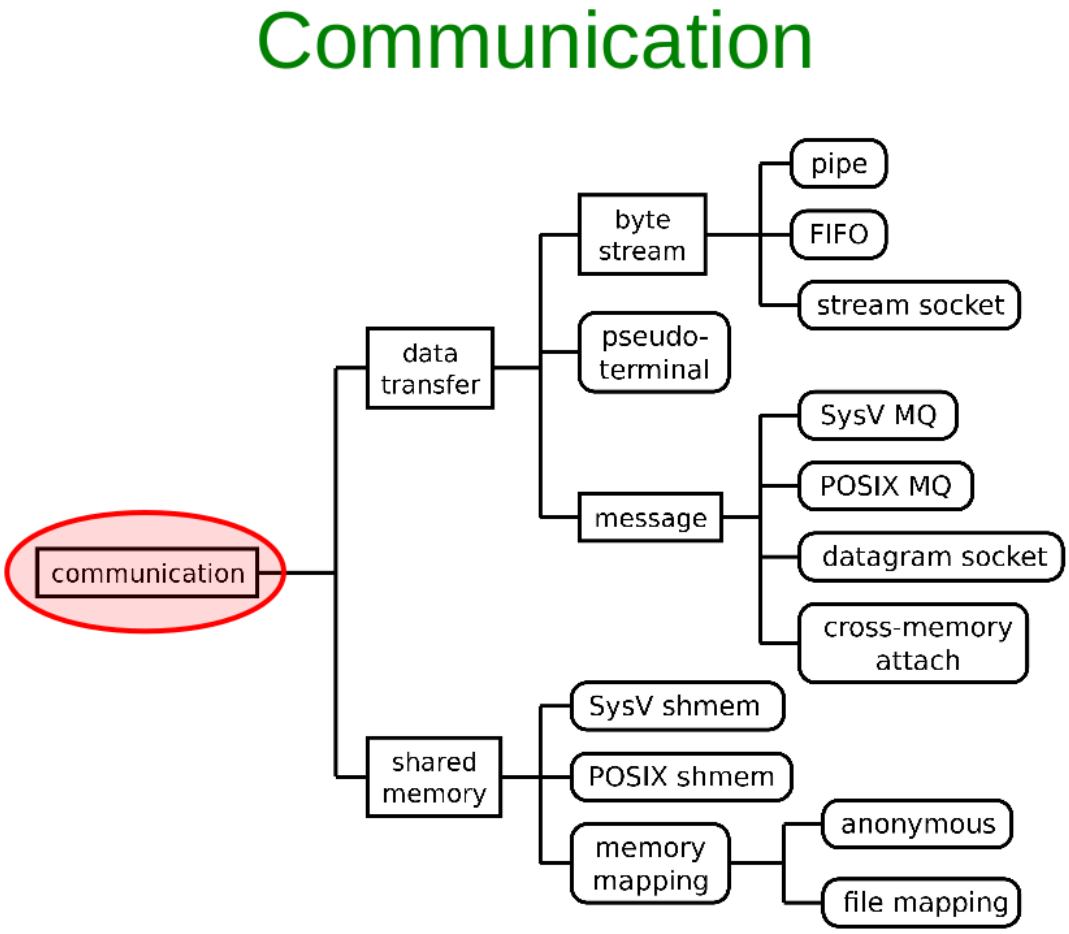
Các cơ chế hỗ trợ trên nền tảng Windows:

- Named Pipe - Ống định danh
- Mailslots
- NetBIOS
- Windows Sockets
- Remote procedure call (RPC)
- Network Dynamic Data Exchange (NetDDE)

Vậy nên chọn cơ chế nào để sử dụng?

Một ứng dụng có thể sử dụng nhiều cơ chế IPC khác nhau. Để có được sự lựa chọn phù hợp, ta phải trả lời các câu hỏi sau:

- Ứng dụng giao tiếp với ứng dụng khác trên một máy tính khác thuộc mạng khác hay chỉ trên cùng máy tính?
- Ứng dụng chạy trên hệ điều hành nào?
- Ứng dụng là một ứng dụng GUI hay là ứng dụng Console? Một vài cơ chế yêu cầu thực hiện trên ứng dụng GUI



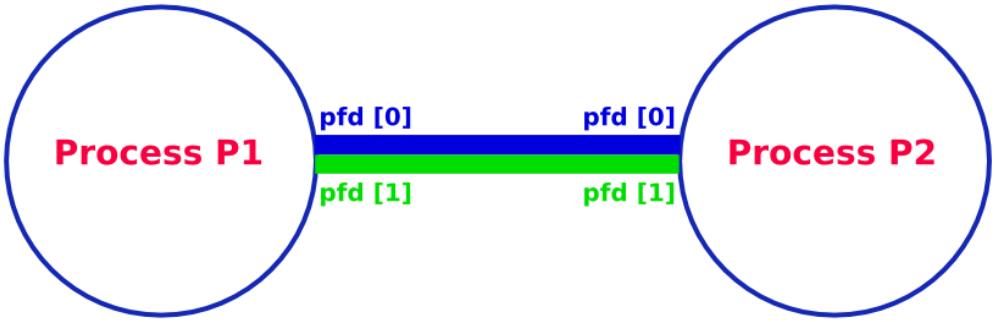
Hình 3. Phân loại IPC

Trong bài nghiên cứu dưới đây, chỉ tìm hiểu về 2 cơ chế: **Pipe** và **Shared Memory (File mapping).**

IV. Pipe

Ống có thể được xem là một trong những cơ chế IPC được sử dụng rộng rãi nhất.

Ống là một **kết nối giữa 2 tiến trình**, đầu ra của tiến trình này trở thành đầu vào của tiến trình khác.



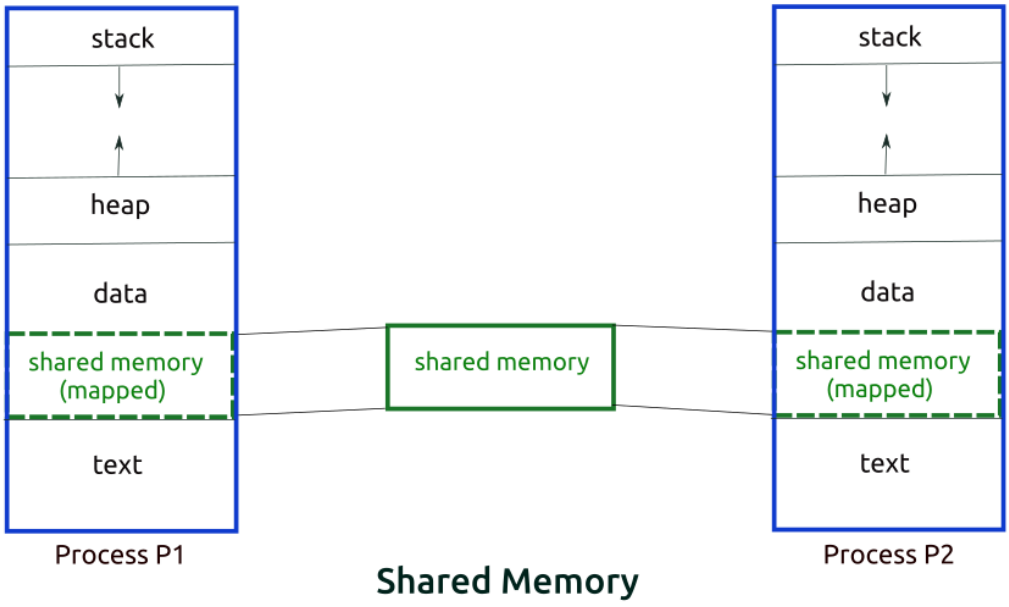
Hình 4. Hai tiến trình kết nối nhau bằng đường ống

Có 2 loại ống cho giao tiếp 2 chiều: ống ẩn danh và ống định danh. Ống ẩn danh cho phép các tiến trình liên quan có thể trao đổi thông tin với nhau. Điển hình, một ống ẩn danh được dùng cho việc chuyển hướng các đầu vào, đầu ra chuẩn của tiến trình con để nó có thể trao đổi dữ liệu với tiến trình cha. Trao đổi dữ liệu 2 chiều, ta phải tạo 2 ống ẩn danh. Tiến trình cha ghi dữ liệu lên 1 ống bằng Write handle trong khi tiến trình con đọc dữ liệu từ ống với Read handle. Ống ẩn danh không thể dùng vượt quá một mạng hay các tiến trình không liên quan đến nhau.

Ống định danh được dùng để truyền dữ liệu giữa các tiến trình không liên quan đến nhau hoặc giữa các tiến trình trên các máy khác nhau.

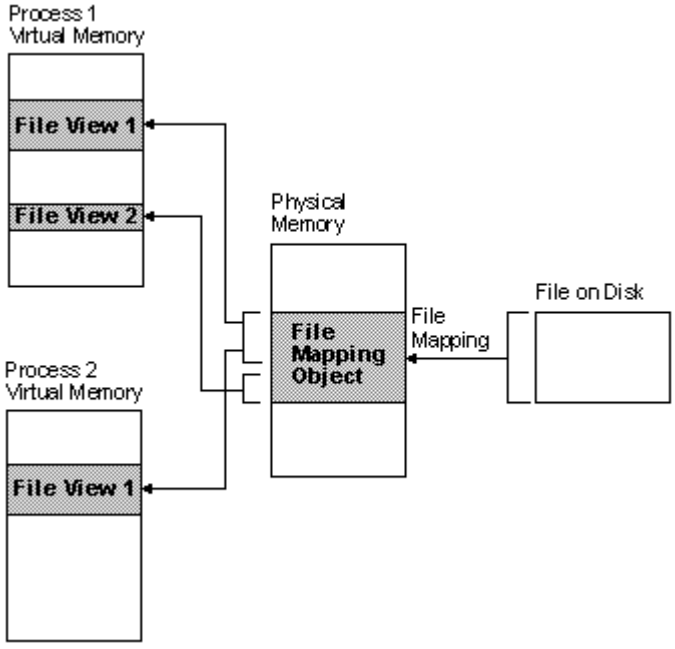
V. Bộ nhớ chia sẻ (Shared Memory)

Bộ nhớ chia sẻ là một trong ba cơ chế IPC có ở Linux và hệ kiểu Unix. Hai cơ chế khác là Hàng đợi thông điệp (Message Queue) và Semaphore. Trường hợp bộ nhớ chia sẻ, một vùng nhớ chia sẻ được tạo ra bởi kernel và kết nối đến vùng dữ liệu của không gian địa chỉ của tiến trình yêu cầu. Tiến trình dùng bộ nhớ chia sẻ giống như các biến toàn cục khác bên trong vùng không gian địa chỉ.



Hình 5. Nguyên lý của Bộ nhớ chia sẻ

Các cơ chế IPC như ống, FIFO, hàng đợi thông điệp (Message Queue) sẽ gửi dữ liệu từ một tiến trình đến tiến trình khác. Thông điệp sẽ được sao chép từ không gian địa chỉ này của tiến trình P1 sang không gian địa chỉ kia của tiến trình P2. Cơ chế bộ nhớ chia sẻ không sao chép như vậy. Tiến trình đầu P1 chỉ đơn giản ghi dữ liệu vào vùng nhớ chia sẻ. Ngay sau ghi xong, dữ liệu tồn tại bên trong tiến trình P2. Vì vậy **cơ chế bộ nhớ chia sẻ là cơ chế nhanh nhất** trong các cơ chế IPC.



Hình 6. Mối quan hệ giữa tệp trên đĩa, đối tượng ánh xạ tệp và Hiển thị tệp

Sử dụng **ánh xạ tệp** (File Mapping), ánh xạ tệp cho phép một tiến trình xem nội dung của 1 tệp như một khối bộ nhớ trong không gian địa chỉ của tiến trình. Tiến trình có thể sử dụng một con trỏ để thực hiện các thao tác đọc và chỉnh sửa nội dung của tệp. Khi 2 hay nhiều tiến trình kết nối cùng ánh xạ tệp, mỗi tiến trình nhận 1 con trỏ, con trỏ này trỏ đến bộ nhớ sở hữu một không gian địa chỉ mà nó có thể sử dụng để đọc và chỉnh sửa nội dung của tệp. Các tiến trình này phải sử dụng một đối tượng đồng bộ như semaphore, để tránh việc xung đột dữ liệu trong môi trường đa tác vụ.

Ta có thể sử dụng một trường hợp đặc biệt của **Ánh xạ tệp** để cung cấp một bộ nhớ chia sẻ giữa các tiến trình

VI. Kí pháp Ba Lan

1. Biểu thức số học

Biểu thức số học là biểu thức nhận được từ các hằng số, biến số và hàm số liên kết với nhau bằng các phép toán số học.

Thứ tự thực hiện: trong ngoặc thực hiện trước ngoài ngoặc thực hiện sau. Đối với biểu thức không có ngoặc thì: **hàm mũ, hàm căn** ưu tiên thực hiện trước sau đó đến **nhân, chia** và **cộng, trừ** thực hiện sau.

Nếu trong biểu thức chứa 1 hằng hay biến kiểu thực thì ta có biểu thức số học thực, giá trị biểu thức cũng thuộc kiểu thực.

2. Định giá biểu thức

- Tính giá trị biểu thức

3. Biểu thức hậu tố

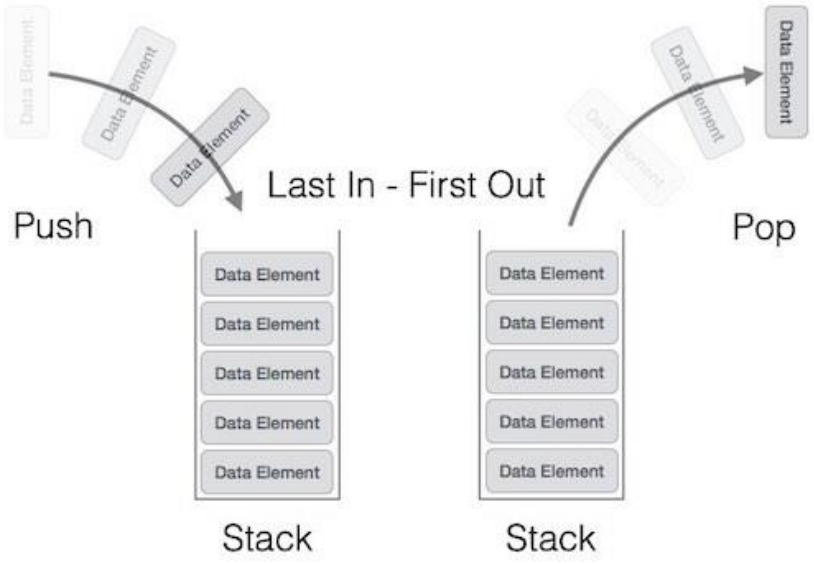
- Biểu thức hậu tố: Biểu thức mà các toán tử sẽ được đặt sau các toán hạng. Cách biểu diễn này được gọi là “ký pháp nghịch đảo Ba Lan” được viết tắt là RPN.

VD : xy+; xy+z-; xyz*+.

4. Cấu trúc ngăn xếp Stack

- Một ngăn xếp là một cấu trúc dữ liệu trừu tượng (Abstract Data Type-ADT), hầu như được sử dụng trong mọi ngôn ngữ lập trình. Đặt tên ngăn xếp bởi vì nó hoạt động như một ngăn xếp trong đời sống thực, ví dụ như một cỗ bài hay một chồng đĩa,...

- Ngăn xếp có cấu trúc dữ liệu dạng LIFO. LIFO là viết tắt của Last-In-First-Out. Ở đây, phần tử được thêm vào cuối cùng sẽ được truy cập đầu tiên. Trong thuật ngữ ngăn xếp hoạt động thêm vào được gọi là hoạt động PUSH, hoạt động lấy ra được gọi là POP.



Hình 7. Hình minh họa cấu trúc ngăn xếp

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

I. Yêu cầu bài toán:

Dùng pipe và shared memory để giao tiếp giữa hai quá trình. Quá trình thứ nhất cho người dùng nhập vào từ bàn phím một chuỗi biểu diễn một phép tính gồm các phần tử +, -, *, /, ^, sqrt, sin, cos, tan.

Ví dụ: 2+12+(12-4-6) *((3+4)-5)

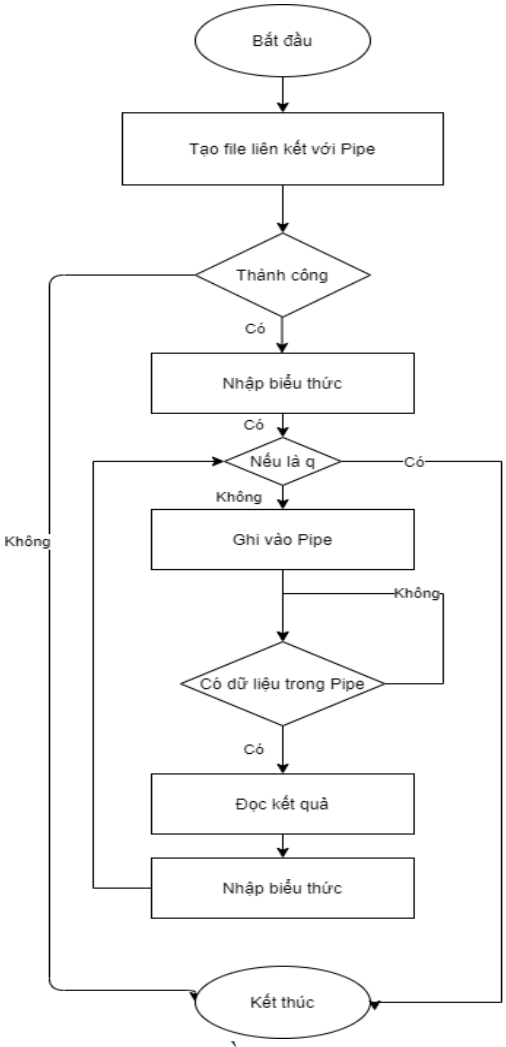
(11+(-2)–((3/4)-5))

Truyền chuỗi dữ liệu này cho quá trình thứ hai. Quá trình thứ hai thực hiện tính

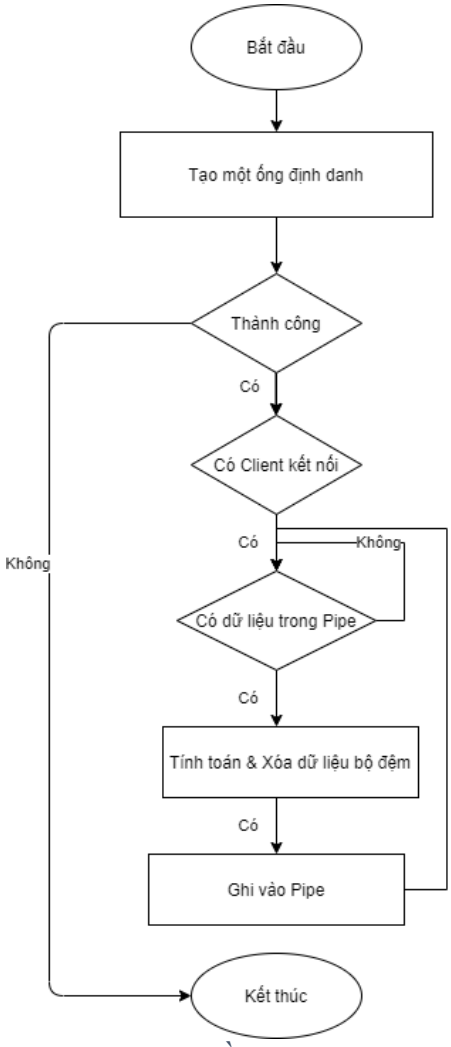
toán trên và trả kết quả về cho quá trình thứ nhất để hiển thị cho người sử dụng biết.

II. Sơ đồ thuật toán

Truyền thông điệp với Pipe

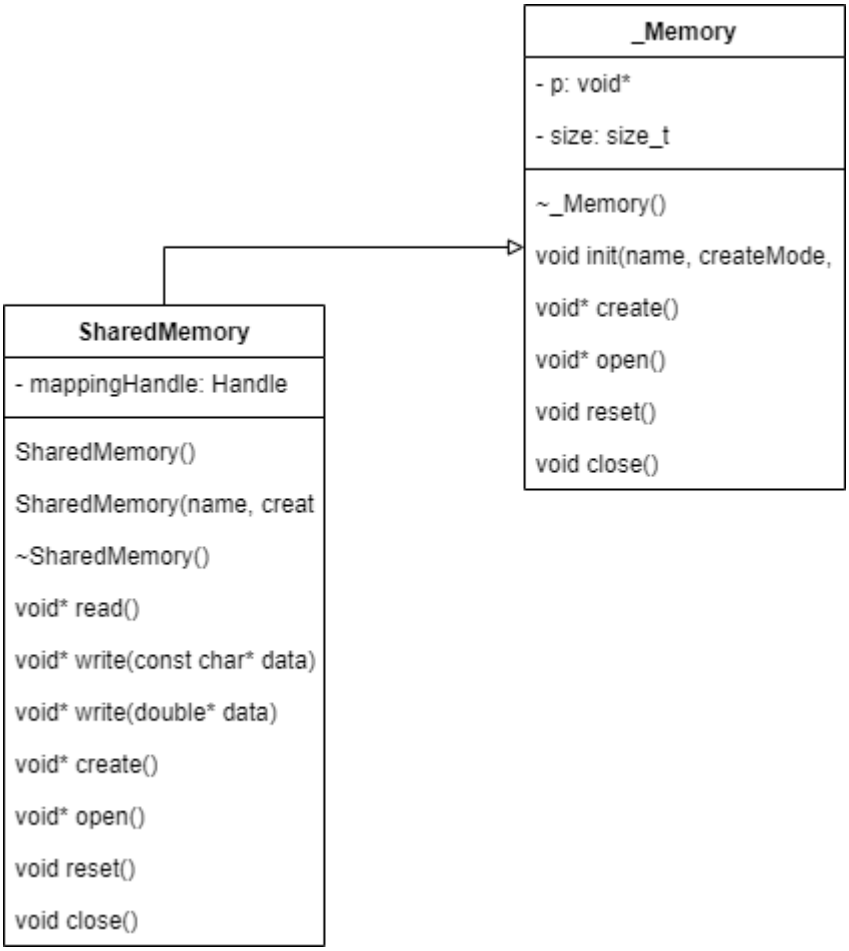


Hình 8. Sơ đồ thuật toán Pipe-Client

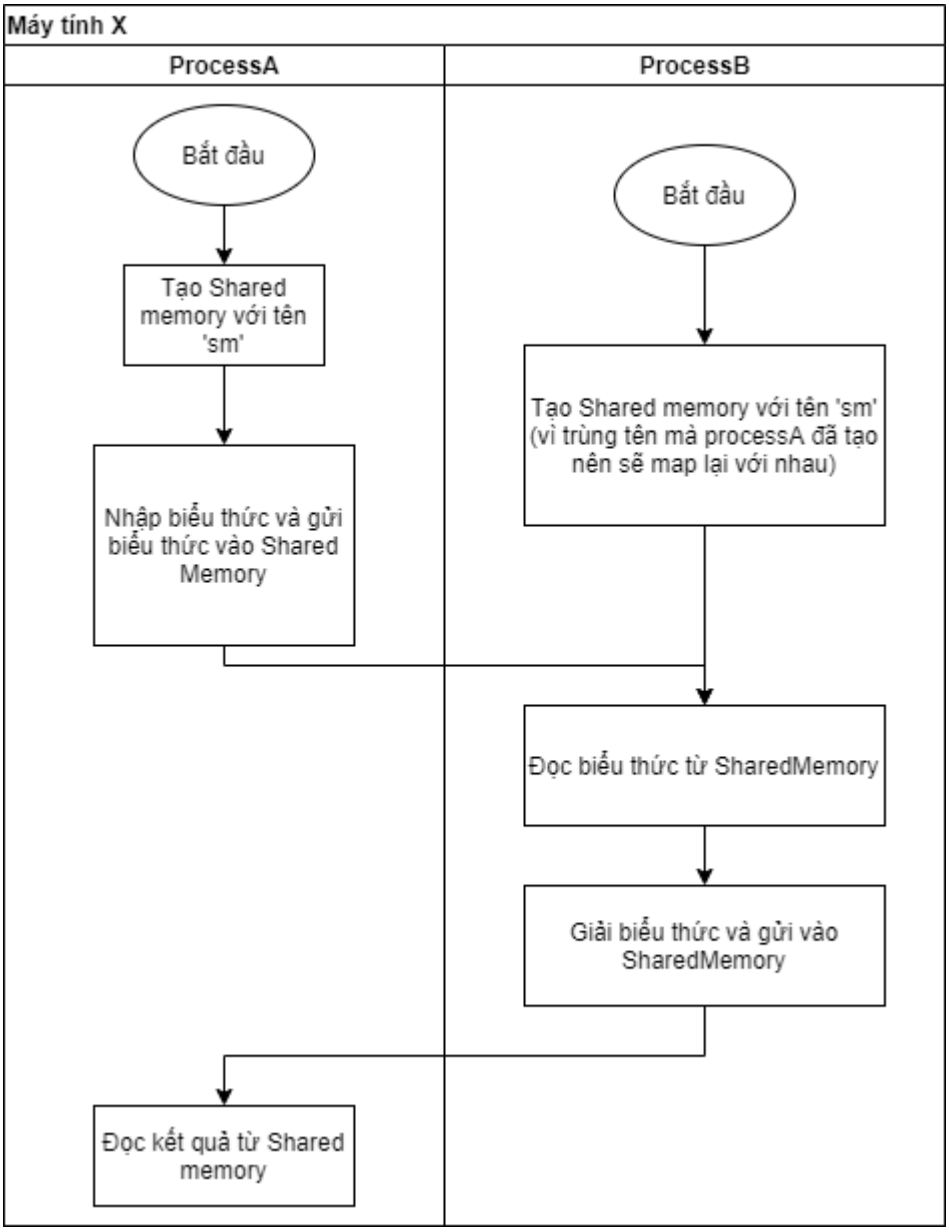


Hình 9. Sơ đồ thuật toán Pipe-Server

Truyền thông điệp với Shared Memory



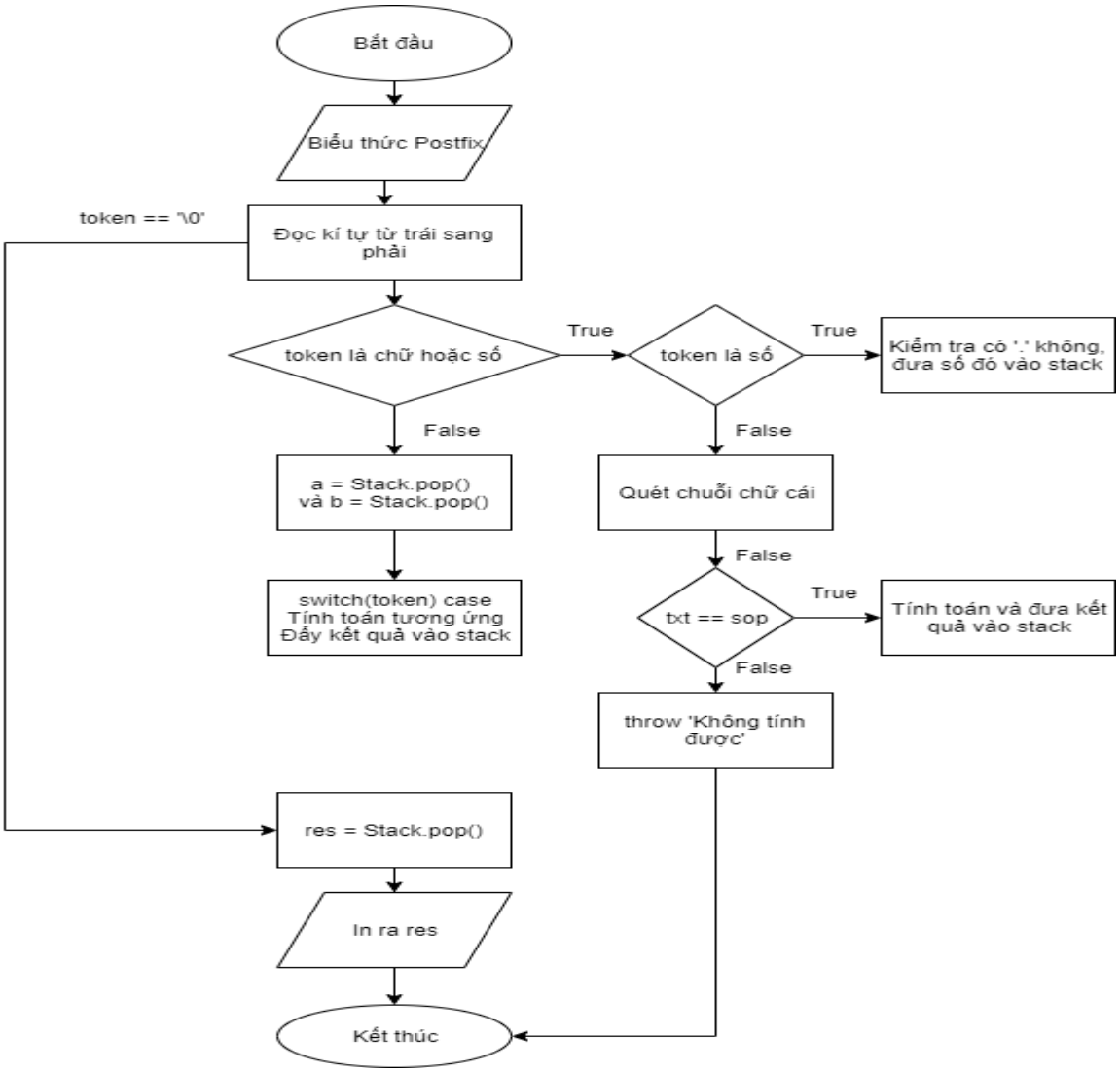
Hình 10. Cấu trúc của Shared Memory



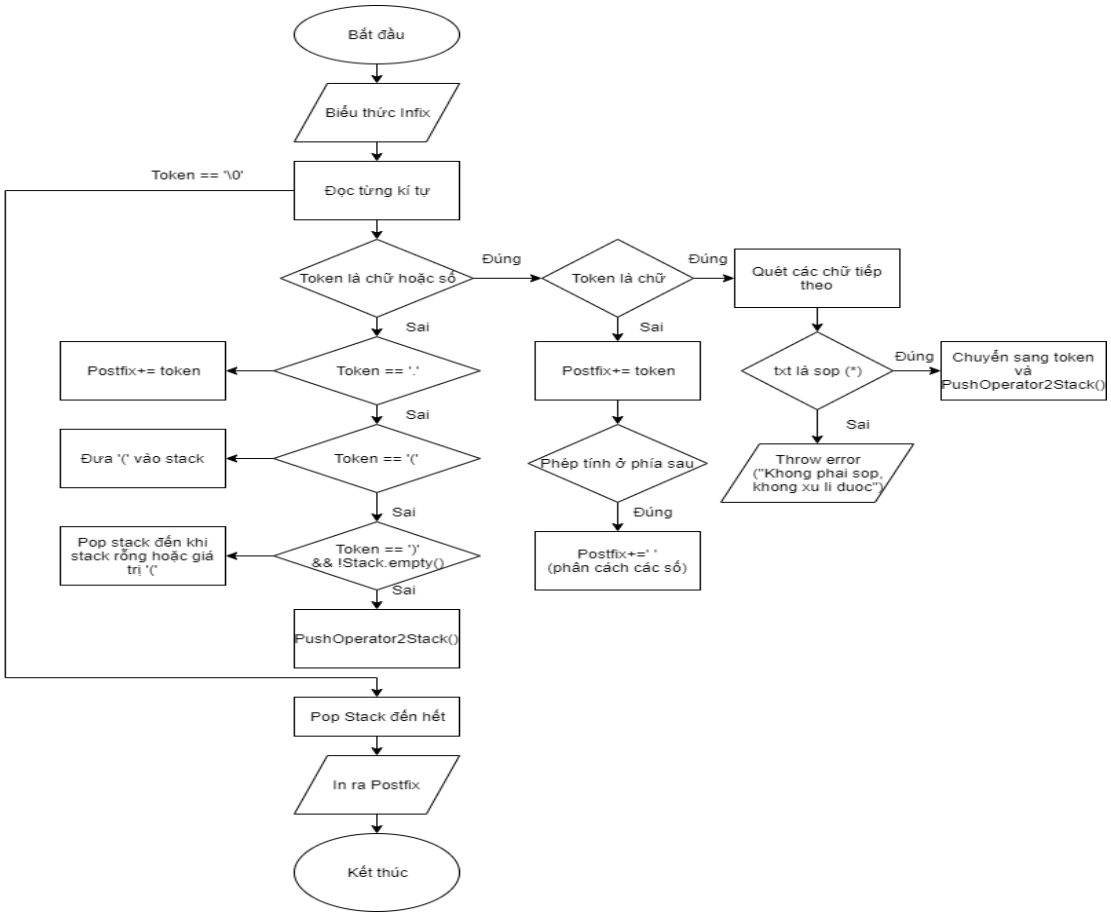
Hình 11. Luồng xử lý của 2 tiến trình trao đổi thông điệp bằng Shared Memory

Tính biểu thức

1.1. Chuyển sang biểu thức hậu tố


Bắt đầu
Biểu thức Postfix
token == '\0'
Đọc kí tự từ trái sang phải
token là chữ hoặc số
True
token là số
True
Kiểm tra có '.' không, đưa số đó vào stack
False
False
a = Stack.pop() và b = Stack.pop()
Quét chuỗi chữ cái
False
switch(token) case Tính toán tương ứng Đẩy kết quả vào stack
txt == sop
True
Tính toán và đưa kết quả vào stack
False
throw 'Không tính được'
res = Stack.pop()
In ra res
Kết thúc

1.2. Định giá biểu thức



III. Môi trường cài đặt

- Ngôn ngữ: C/C++;
- Công cụ hỗ trợ:
  + IDE: Visual Studio Code

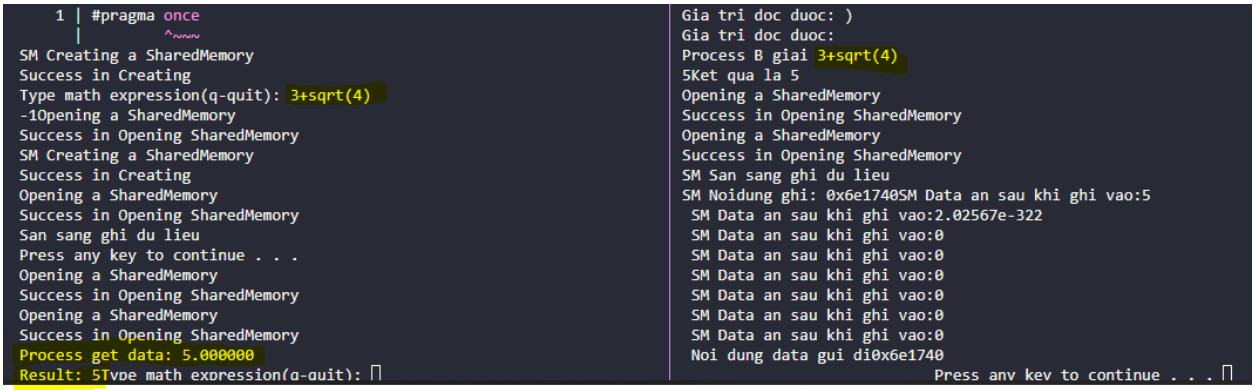
CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ

I. Demo chương trình

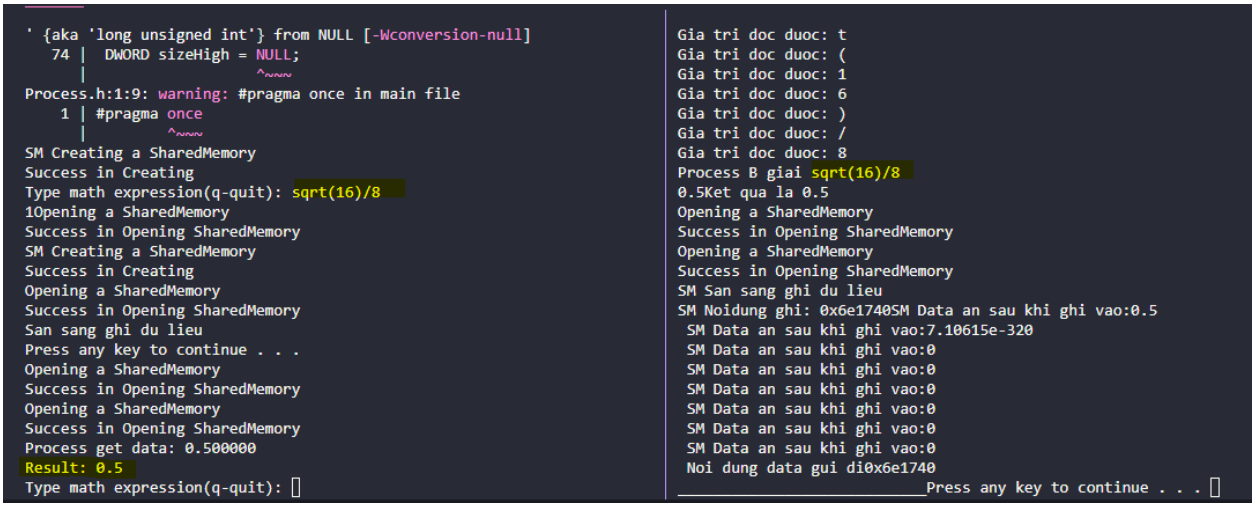
Pipe

Hình 12. Demo chương trình sử dụng Pipe để truyền thông điệp

Shared Memory



Hình 13. Demo chương trình sử dụng Shared Memory để truyền thông điệp 1



Hình 14. Demo chương trình sử dụng Shared Memory để truyền thông điệp 2

II. Kết luận và hướng phát triển

Kết luận:

- Các cơ chế giao tiếp giữa các tiến trình là mang lại nhiều lợi ích trong việc phát triển ứng dụng ngày nay. Hỗ trợ lập trình viên trong việc phát triển nhanh hơn các ứng dụng đa tiến trình mà không cần can thiệp sâu vào nhân hệ điều hành.

Hướng phát triển:

- Tìm hiểu những cơ chế giao tiếp giữa các tiến trình còn lại
- Phát triển giao diện cho ứng dụng

KẾT LUẬN CHUNG

TÀI LIỆU THAM KHẢO

- https://docs.microsoft.com/en-us/windows/win32/ipc/interprocess-communications?redirectedfrom=MSDN
- https://www.stdio.vn/articles/ung-dung-stack-bieu-thuc-hau-to-postfix-471
- https://codelearn.io/sharing/lap-trinh-socket-co-voi-tcpip-ava