CONCURRENCY PROGRAMMING

Contents

[1. How Does a Program Start? 2](#_Toc139987278)

[2. Process and Thread – How are these different? 2](#_Toc139987279)

[2.1. What is Process? 2](#_Toc139987280)

[2.2. What is Thread? 2](#_Toc139987281)

[2.3. Differences between Process and Thread? 2](#_Toc139987282)

[3. Multiprocessing 2](#_Toc139987283)

[3.1. Advantages of multiprocessing 2](#_Toc139987284)

[3.2. Disadvantages of multiprocessing 2](#_Toc139987285)

[3.3. Multiprocessing in C++ 2](#_Toc139987286)

[3.3.1. fork() 2](#_Toc139987287)

[3.3.2. Data of process 2](#_Toc139987288)

[3.4. Concurrency/Data sharing in multiprocessing 3](#_Toc139987289)

[3.4.1. Shared memory 3](#_Toc139987290)

[3.4.2. Socket 3](#_Toc139987291)

[3.4.3. Named Pipe 3](#_Toc139987292)

[4. Multithreading 3](#_Toc139987293)

[4.1. Advantages of multithreading 3](#_Toc139987294)

[4.2. Disadvantages of multithreading 3](#_Toc139987295)

[4.3. Multithreading in C++ 3](#_Toc139987296)

[4.4. Concurrency in multithreading 4](#_Toc139987297)

[4.4.1. <mutex> 4](#_Toc139987298)

[4.4.2. <condition\_variable> 4](#_Toc139987299)

[4.4.3. Semaphore concept 4](#_Toc139987300)

# How Does a Program Start?

1. Biên dịch hoặc Diễn giải: Mã của chương trình được chuyển đổi thành các hướng dẫn mà máy có thể đọc được.
2. Đang tải: Hệ điều hành phân bổ bộ nhớ và tải tệp thực thi của chương trình, bao gồm các hướng dẫn và dữ liệu của nó, vào bộ nhớ.
3. Khởi tạo: Chương trình thiết lập trạng thái ban đầu của nó bằng cách phân bổ bộ nhớ, thiết lập kết nối, khởi tạo các biến và chuẩn bị bất kỳ tài nguyên cần thiết nào.
4. Chức năng chính: Chương trình bắt đầu thực hiện từ một điểm vào được chỉ định, thường là một chức năng chính, đóng vai trò là điểm bắt đầu cho việc thực hiện chương trình.
5. Thực thi: Chương trình thực hiện các lệnh của nó một cách tuần tự, thực hiện các tính toán, tương tác với các thiết bị đầu vào/đầu ra và thực hiện các tác vụ mong muốn được xác định trong mã của nó.
6. Chấm dứt: Chương trình kết thúc hoặc gặp một câu lệnh thoát rõ ràng, báo hiệu sự hoàn thành của nó. Nó có thể giải phóng tài nguyên được phân bổ, đóng tệp hoặc kết nối và thực hiện các tác vụ dọn dẹp trước khi chấm dứt.

# Process and Thread – How are these different?

[]

## What is Process?

[write 2-5 lines]

Quá trình trong lập trình đồng thời là một phiên bản của chương trình được thực thi đồng thời. Các quá trình chạy độc lập với không gian bộ nhớ riêng và tài nguyên hệ thống riêng của chúng. Chúng cung cấp sự cô lập và cho phép thực hiện song song nhiều nhiệm vụ. Các quá trình giao tiếp thông qua các cơ chế giao tiếp giữa các quá trình (IPC). Lập lịch quá trình do hệ điều hành quyết định thứ tự thực thi và phân bổ tài nguyên.

## What is Thread?

[write 2-5 lines]

Thread trong lập trình đồng thời là một đơn vị thực thi nhẹ trong một quá trình. Các luồng chạy song song với các luồng khác trong cùng một quá trình. Chúng chia sẻ cùng không gian bộ nhớ, mô tả tệp và tài nguyên hệ thống. Luồng cho phép song song hóa và cho phép thực hiện các nhiệm vụ đồng thời. Các cơ chế đồng bộ hóa luồng được sử dụng để điều phối truy cập vào các tài nguyên chung giữa các luồng.

## Differences between Process and Thread?

… [Find at least 5 differences]

|  |  |
| --- | --- |
| **Process** | **Thread** |
| Đơn vị thực thi độc lập | Một phần của một process |
| Có không gian bộ nhớ riêng | Chia sẻ không gian bộ nhớ với các thread cùng process |
| Yêu cầu giao tiếp giữa các process (IPC) để chia sẻ dữ liệu | Có thể truy cập trực tiếp vào bộ nhớ chung |
| Có file descriptor riêng | Chia sẻ file descriptor với các thread cùng process |
| Tạo và chấm dứt process yêu cầu nhiều công việc hơn | Tạo và chấm dứt thread nhanh hơn và ít tốn kém hơn |
| Cung cấp cô lập lỗi tốt hơn | Một lỗi trong một thread có thể làm chấm dứt toàn bộ process |
| Có thể chạy trên các bộ xử lý khác nhau và tận dụng đa xử lý | Chạy trên cùng một bộ xử lý của process |
| Yêu cầu cơ chế IPC rõ ràng để giao tiếp và đồng bộ | Giao tiếp và đồng bộ dễ dàng hơn thông qua bộ nhớ chung |

# Multiprocessing

## Advantages of multiprocessing

[Ưu điểm 1]: Tăng hiệu suất

Multiprocessing cho phép thực thi nhiều quy trình cùng một lúc, cải thiện hiệu suất tổng thể.

Ví dụ: Chạy đồng thời nhiều luồng của một chương trình mã hóa video để mã hóa video nhanh hơn.

[Ưu điểm 2]: Tận dụng tài nguyên tốt hơn

Multiprocessing tối ưu hóa việc sử dụng tài nguyên như thời gian CPU và bộ nhớ bằng cách phân phối nhiệm vụ cho nhiều quy trình.

Ví dụ: Chạy các quy trình song song để xử lý các tập dữ liệu lớn, tận dụng hiệu quả tất cả các lõi CPU có sẵn.

[Ưu điểm 3]: Tăng tính chịu lỗi

Multiprocessing cải thiện độ tin cậy hệ thống vì lỗi trong một quy trình không ảnh hưởng đến các quy trình khác.

Ví dụ: Trong một máy chủ web, nếu một quy trình gặp lỗi, các quy trình khác vẫn có thể tiếp tục phục vụ yêu cầu.

## Disadvantages of multiprocessing

[Nhược điểm 1]: Tăng độ phức tạp

Multiprocessing làm tăng độ phức tạp do yêu cầu giao tiếp giữa các quy trình, đồng bộ hóa và quản lý tài nguyên.

Ví dụ: Đồng bộ hóa việc chia sẻ dữ liệu giữa nhiều quy trình trong khi tránh các tình huống đua nhau hoặc mắc kẹt.

[Nhược điểm 2]: Gánh nặng

Quản lý nhiều quy trình tạo ra gánh nặng về việc chuyển đổi ngữ cảnh, quản lý bộ nhớ và giao tiếp giữa các quy trình.

Ví dụ: Chuyển đổi ngữ cảnh giữa các quy trình tạo ra gánh nặng khi hệ thống cần lưu và khôi phục trạng thái của mỗi quy trình.

## Multiprocessing in C++

### fork()

[Example]

[Return value of fork()]

### Data of process

[Demo how processes access its data.]

Trong quá trình quản lý dữ liệu, một quá trình có thể sử dụng và quản lý dữ liệu theo hai loại chính: biến toàn cục và dữ liệu được sử dụng trong suốt vòng đời của quá trình.

Biến toàn cục:

Biến toàn cục có thể được truy cập và sử dụng bởi tất cả các quá trình trong hệ thống.

Các quá trình có thể đọc và ghi vào biến toàn cục, cho phép chia sẻ dữ liệu.

Biến toàn cục được lưu trữ trong bộ nhớ chia sẻ, mà các quá trình sử dụng cùng một vùng nhớ chia sẻ có thể truy cập.

Dữ liệu sử dụng trong suốt vòng đời:

Đây là dữ liệu được tạo và sử dụng bởi một quá trình trong quá trình hoạt động của nó.

Bao gồm các biến cục bộ cụ thể cho mỗi quá trình, có phạm vi và quyền truy cập hạn chế.

Mỗi quá trình có không gian bộ nhớ riêng để lưu trữ dữ liệu này.

[Create a graph/diagram to demo how processes manage data]

## Concurrency/Data sharing in multiprocessing

### Shared memory

[Concept – 2-5 lines]

Bộ nhớ chia sẻ là một phương pháp truyền thông giữa các quá trình (IPC) trong đó nhiều quá trình có thể truy cập vào cùng một vùng nhớ. Điều này cho phép các quá trình chia sẻ dữ liệu mà không cần truyền tin nhắn tường minh.

[Example code]

Trong ví dụ mã trên, đoạn mã tạo ra một đoạn nhớ chia sẻ sử dụng hàm shmget() với kích thước 1024 byte và khóa là 1234. Hàm shmat() được sử dụng để gắn đoạn nhớ chia sẻ vào quá trình, sau đó dữ liệu có thể được ghi vào đoạn nhớ chia sẻ bằng cách sử dụng các phương pháp tiêu chuẩn như sprintf(). Cuối cùng, hàm shmdt() được sử dụng để tách đoạn nhớ chia sẻ khỏi quá trình

[Output of example code]

Đoạn mã này tạo ra một đoạn nhớ chia sẻ và ghi chuỗi "Hello, shared memory!" vào đó. Các quá trình khác có thể truy cập đoạn nhớ chia sẻ này bằng cách gắn vào cùng một đoạn nhớ chia sẻ.

### Socket

[Concept – 2-5 lines]

Socket là một phương pháp IPC (Inter-Process Communication) cho phép giao tiếp giữa các quá trình qua mạng hoặc cùng một máy tính. Các quá trình có thể gửi và nhận dữ liệu thông qua socket.

[Example code]

Trong ví dụ mã trên, đoạn mã tạo ra một socket sử dụng hàm socket() với các tham số AF\_INET và SOCK\_STREAM. Sau đó, thông tin địa chỉ của server được thiết lập trong cấu trúc sockaddr\_in. Hàm bind() được sử dụng để gắn socket vào cổng 8080, và hàm listen() được sử dụng để lắng nghe kết nối đến socket.

Khi một kết nối mới được thiết lập, hàm accept() chấp nhận kết nối và tạo ra một socket mới (new\_socket) để giao tiếp với client. Sau đó, dữ liệu được đọc từ socket sử dụng hàm read(), và một tin nhắn "Hello from server" được gửi lại cho client sử dụng hàm send().

[Output of example code]

### Named Pipe

[Concept – 2-5 lines]

Named pipe (ống tên) còn được gọi là FIFO (First-In-First-Out) là một hình thức giao tiếp giữa các quá trình bằng cách ghi và đọc dữ liệu vào và ra khỏi một tệp ống chung.

[Example code]

[Output of example code]

Trong ví dụ mã trên, đoạn mã tạo ra một named pipe (fifo) sử dụng hàm mkfifo() và đặt tên là "named\_pipe\_example". Sau đó, hàm open() được sử dụng để mở named pipe để ghi dữ liệu vào nó.

Sau khi named pipe được mở, dữ liệu có thể được ghi vào named pipe bằng cách sử dụng hàm write(), như trong ví dụ trên dữ liệu "Hello, named pipe!" được ghi vào named pipe. Cuối cùng, hàm close() được sử dụng để đóng named pipe.

Các quá trình khác có thể mở cùng một named pipe để đọc dữ liệu từ nó bằng cách sử dụng hàm open() với chế độ đọc. Điều này cho phép các quá trình giao tiếp với nhau thông qua named pipe bằng cách ghi và đọc dữ liệu từ tệp ống chung.

# Multithreading

## Advantages of multithreading

[Lợi ích 1]: Xử lý và phản hồi thời gian thực

Multithreading cho phép xử lý và phản hồi thời gian thực trong các ứng dụng yêu cầu cập nhật ngay lập tức hoặc liên tục.

Ví dụ: Phần mềm sản xuất âm nhạc sử dụng multithreading để xử lý âm thanh thời gian thực trong khi cung cấp giao diện người dùng phản hồi.

[Lợi ích 2]: Thiết kế đơn giản và tính module hóa

Multithreading hỗ trợ phương pháp thiết kế module, chia nhỏ các nhiệm vụ phức tạp thành các luồng nhỏ, đơn giản hóa mã nguồn và cải thiện khả năng bảo trì.

Ví dụ: Ứng dụng xử lý ảnh với các luồng riêng biệt cho việc tải, lọc và lưu ảnh.

## Disadvantages of multithreading

[Khuyết điểm 1]: Độ phức tạp của đồng bộ hóa

Multithreading đưa ra yêu cầu về các cơ chế đồng bộ hóa để quản lý tài nguyên chung, điều này có thể gây ra sự phức tạp và các vấn đề về đồng bộ hóa.

Ví dụ: Đồng bộ việc truy cập vào một kết nối cơ sở dữ liệu chung trong một ứng dụng đa luồng, điều này yêu cầu đồng bộ cẩn thận để tránh sự không nhất quán dữ liệu.

[Khuyết điểm 2]: Khó khăn trong việc gỡ lỗi và kiểm thử

Việc gỡ lỗi và kiểm thử các ứng dụng đa luồng có thể gặp khó khăn do tính không xác định của thực thi luồng.

Ví dụ: Giải quyết các tình huống cạnh tranh (race conditions) trong một ứng dụng giao dịch tài chính đa luồng, nơi tương tác giữa các luồng phụ thuộc vào thời gian có thể khó tái hiện và gỡ lỗi.

## Multithreading in C++

[Example with no concurrency]

[Output of example]

## Concurrency in multithreading

### <mutex>

**[Concept – 2-5 lines]**

**[Example code]**

**[Output of example code]**

### <condition\_variable>

[Concept – 2-5 lines]

[Example code]

[Output of example code]

### Semaphore concept

[Concept – 2-5 lines]

[Example code]

[Output of example code]