OPERATING SYSTEM

LAB 03 | LÀM VIỆC VỚI CÁC TIẾN TRÌNH TRONG C

Đã đến lúc làm việc với các tiến trình trong C, trong bài LAB này, người học sẽ tìm hiểu cách tạo và thực hiện giao tiếp liên tiến trình

MỤC TIÊU

Sau LAB này, sinh viên sẽ có thể:

1. Tạo tiến trình theo ba cách: sử dụng hàm fork, exec, và system
2. Thực hiện giao tiếp giữa các quá trình thông qua cơ chế message passing

NỘI DUNG

CHUẨN BỊ

1. Giới thiệu về Google Colab

NỘI DUNG

THỰC HÀNH

1. Tạo tiến trình
2. Giao tiếp giữa các tiến trình

GIỚI THIỆU VỀ GOOGLE COLAB

Chuẩn bị

00.

0. Giới thiệu về Google Colab

Google Colab là gì?

- Môi trường Jupyter Notebook dựa trên đám mây
- Cho phép bạn viết và thực thi mã Python và lệnh shell
- Cung cấp quyền truy cập miễn phí vào GPU và TPU để tính toán tăng tốc

0. Giới thiệu về Google Colab

Các tính năng chính:

Môi trường hợp tác

Chia sẻ sổ ghi chép và cộng tác trong thời gian thực với người khác

Thư viện được cài đặt sẵn

Đi kèm với nhiều thư viện phổ biến như TensorFlow, Keras, PyTorch, v.v.

Tích hợp lưu trữ đám mây

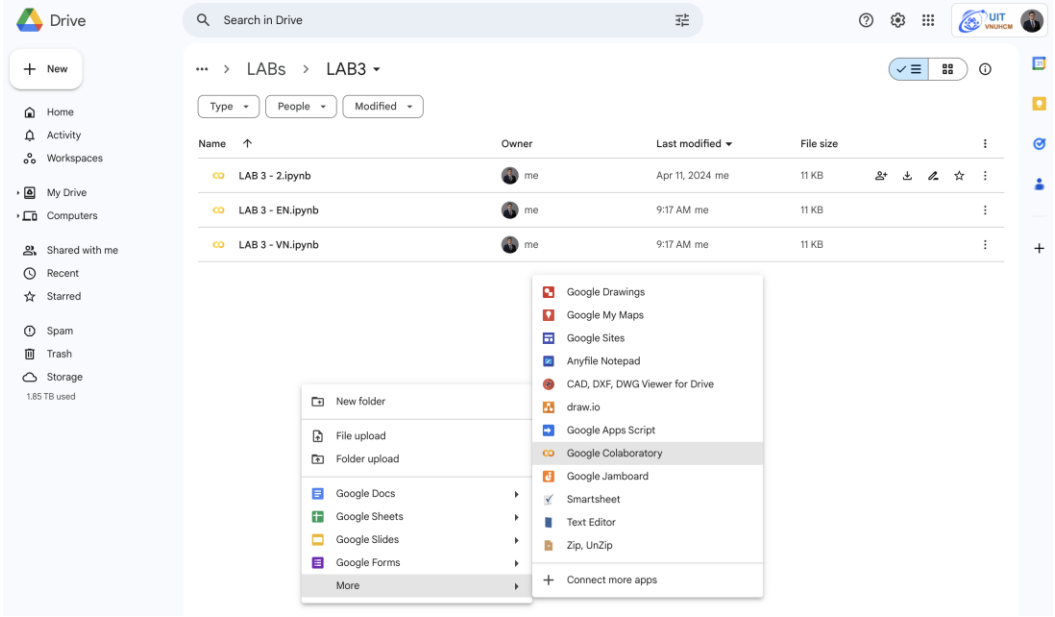
Dễ dàng lưu và tải tệp từ Google Drive

0. Giới thiệu về Google Colab

Bắt đầu với:

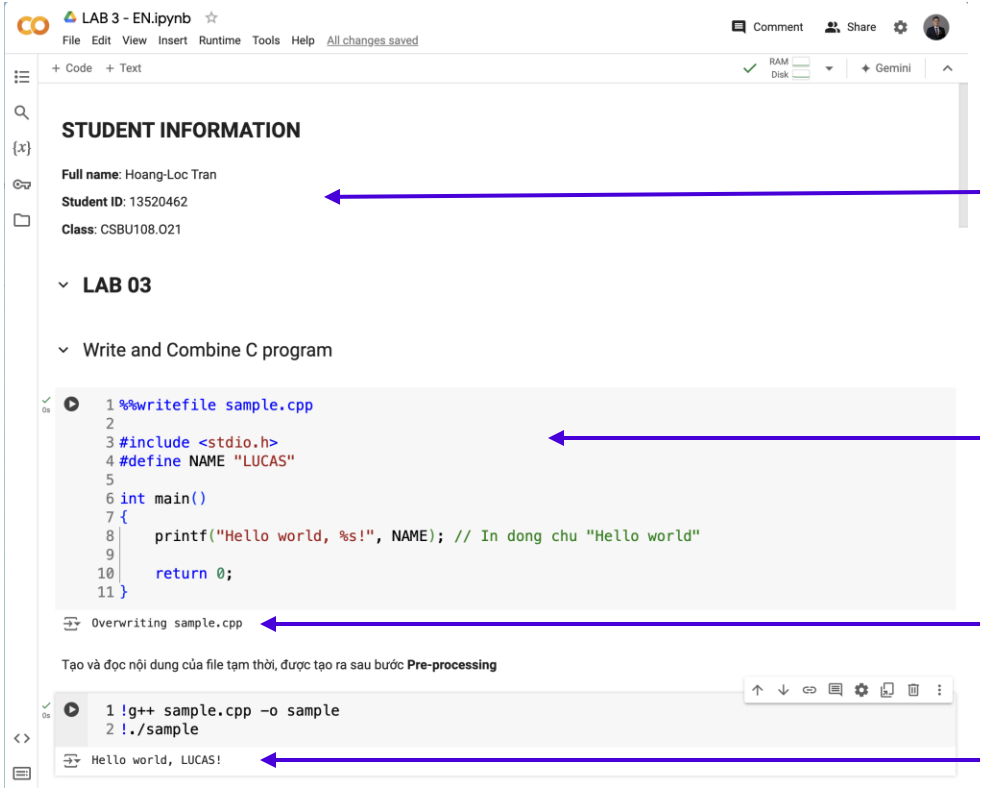
Truy cập: **colab.research.google.com**

Hoặc bạn có thể trực tiếp tạo Google Colab Notebook trong Google Drive của mình



0. Giới thiệu về Google Colab

Mẫu



Văn bản markdown

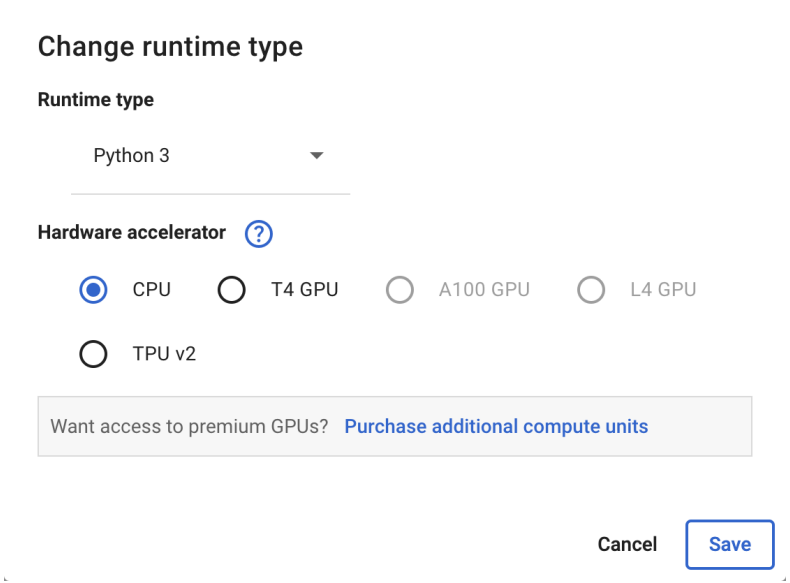
Mã

Đầu ra

Đầu ra

0. Giới thiệu về Google Colab

Thay đổi thời gian chạy

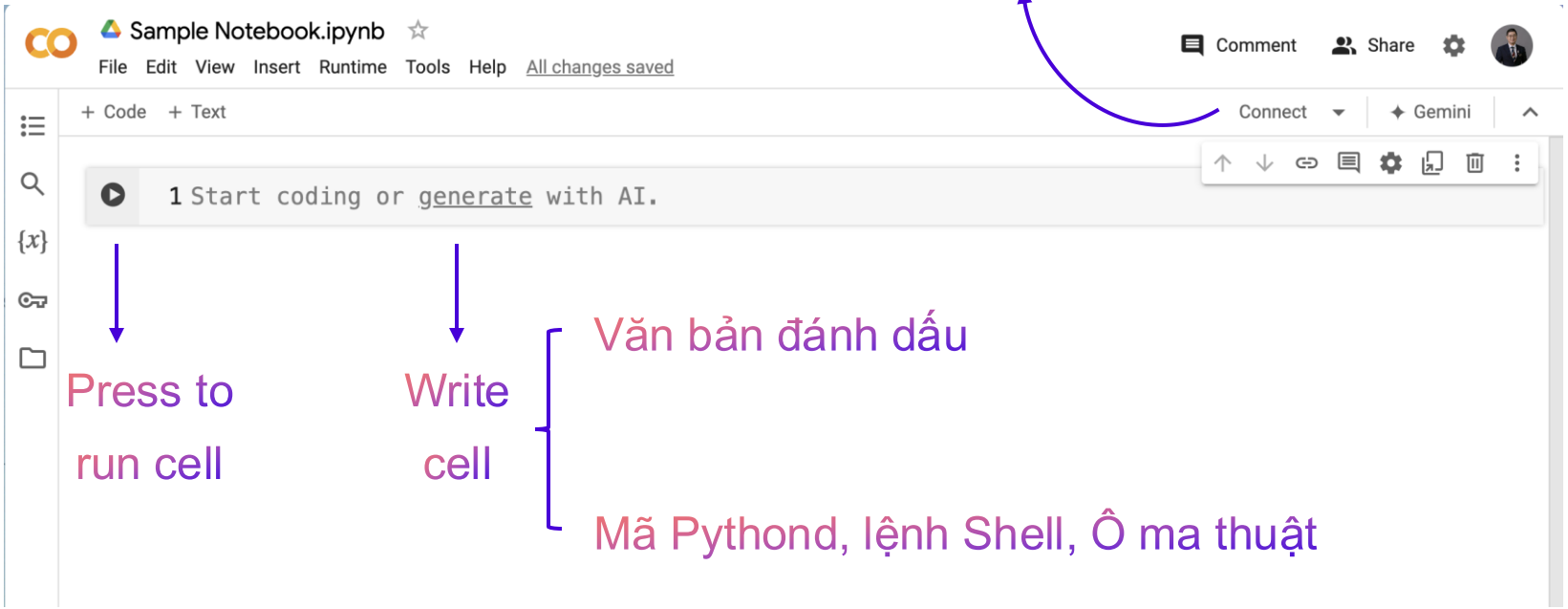


Bạn có thể thay đổi loại thời gian chạy của mình bằng cách chọn

Menu Runtime → Runtime type

0. Giới thiệu về Google Colab

GUI



0.1. Văn bản markdown

Markdown là gì?

- Ngôn ngữ markdown gọn nhẹ
- Được sử dụng để định dạng văn bản với cú pháp văn bản thuần túy
- Thường được sử dụng trong sổ ghi chép Jupyter để làm tài liệu

0.1. Văn bản markdown

Cú pháp markdown cơ bản

Đánh dấu	Cách sử dụng	Kết quả	Ghi chú
#	# Header 1 ## Header 2	**Header 1** Header 2	Headers are collapsible
*	*Emphasis*	*Emphasis*	
**	**Bold**	**Bold**	

0.1. Văn bản markdown

Cú pháp markdown cơ bản

Đánh dấu	Cách sử dụng	Kết quả	Ghi chú
`	This is `Inline code`	This is Inline code	
``	``` # Code block print(“Hello world!” ```	# Code block print (”Hello world”)	

0.1. Văn bản markdown

Cú pháp markdown cơ bản

Đánh dấu	Cách sử dụng	Kết quả	Ghi chú
1.	1. Item 1	1. Item	Ordered list
*	* Item 1	• Item 1	Unordered list
[Link Text](URL)	[Google](https://www.google.com)	Google	Embedded link

0.1. Văn bản markdown

Cú pháp markdown cơ bản

Đánh dấu	Cách sử dụng	Kết quả	Ghi chú
![Alt Text](Image URL)	![Logo](https://www.uit.edu.vn/sites/vi/files/resize/images/Logos/Logo_UIT_Web_Transparent-218x261.png)		

0.1. Văn bản đánh dấu

Ví dụ

```
![Logo](https://www.uit.edu.vn/sites/vi/files/resize/images/Logos/Logo_UIT_Web_Transparent-218x261.png)

#**STUDENT INFORMATION**

**Full name**: Hoang-Loc Tran

**Student ID**: 13520462

**Class**: CSBU108.O21
```

STUDENT INFORMATION

Full name: Hoang-Loc Tran

Student ID: 13520462

Class: CSBU108.O21

Bạn cũng có thể sử dụng thanh công cụ này để tạo markdown

0.2. Mã

Mã Python

```
[7]  1 print ("Hello world")
```

```
Hello world
```

Lệnh Shell

```
[8]  1 !ls /content
```

```
sample  sample.cpp  sample_data
```

Magic cell

```
1 %%writefile hello.cpp
2 #include <stdio.h>
3
4 int main()
5 {
6     printf("Hello world!");
7     return 0;
8 }
```

```
Writing hello.cpp
```

0.2. Mã

Lệnh Shell

- Bạn có thể thực hiện lệnh shell trong Google Colab giúp bạn interract với máy chủ
- **Hãy nhớ thêm dấu chấm than (!) trước lệnh**

```
!chmod 755 script.sh
```

0.2. Code

Magic cell

- Trong Google Colab, **“magic cell”** đề cập đến việc sử dụng các lệnh ma thuật IPython. Đây là những lệnh đặc biệt có tiền tố với một hoặc hai dấu hiệu phần trăm (**%** hoặc **%%**) cung cấp một cách tốc ký để thực hiện các nhiệm vụ khác nhau
- Trong chủ đề này, chúng tôi chỉ sử dụng**%%writefile** để tạo mã nguồn CPP.

0.2. Mã

Magic cell

0s

```
%%writefile hello.cpp
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

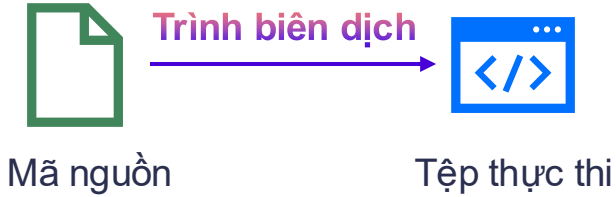
```
Writing hello.cpp
```

Magic cell này tạo file

hello.cpp

0.2. Mã

Biên dịch và chạy chương trình C++



Kết hợp nguồn C

```
!gcc source.c –o output_file
```

Kết hợp nguồn C++

```
!g++ source.cpp –o output_file
```

Nếu bạn sử dụng máy ảo local thay vì Google Colab, hãy nhớ xóa(!) trong lệnh

0.2. Mã

Biên dịch và chạy chương trình C++

Chạy chương trình

```
!path_to_output_file
```

Ví dụ

```
!./output
```

```
!/content/output
```

Nếu bạn sử dụng máy ảo local thay vì Google Colab, hãy nhớ xóa(!) trong lệnh

TẠO TIẾN TRÌNH

1.1. Tạo tiến trình với fork()

01.

1.1. Tạo quy trình vớifork()

Bài tập trên lớp: 01

Phần đọc và triển khai

Phần: **1.1. Tạo quy trình với fork()**

Trả lời các câu hỏi dưới đây:

a. Mô tả cách thức **fork()** hoạt động?

b. Đặc điểm của các quy trình được tạo bởi **fork()**?

- Quan hệ với quá trình ban đầu? PID? Mã nguồn? Làm thế nào chúng ta có thể xác định / phân công công việc cho họ?

TẠO TIẾN TRÌNH

1.2. Tạo tiến trình với exec()

01.

1.2. Tạo tiến trình với exec()

- Các hàm thuộc họ *exec có chức năng* **thay thế** toàn bộ mã và dữ liệu của tiến trình đang chạy. Nó **Không** tạo ra một quy trình mới; Thay vào đó, nó **biến** quy trình hiện có thành một quy trình mới.

1.2. Tạo quy trình với exec()

Bài tập trên lớp: 02

Phần đọc và triển khai

Phần: **1.2. Tạo quy trình với exec()**

Trả lời các câu hỏi dưới đây:

a. Giải thích cú pháp của hàm **execlp()** ?

b. Đặc điểm của các quy trình được tạo bởi **exec()**?

- Quan hệ với quá trình ban đầu? PID? Mã nguồn? Làm thế nào chúng ta có thể xác định / phân công công việc cho họ?

TẠO TIẾN TRÌNH

1.3. Tạo quy trình với system()

01.

1.3. Tạo tiến trình với system()

- **system()** Thực hiện một lệnh shell được chỉ định dưới dạng một chuỗi.
  - Nó phân nhánh một quá trình con
  - Chạy lệnh shell
  - Chờ lệnh kết thúc
  - Rồi quay trở lại.

1.3. Tạo tiến trình với system()

Bài tập trên lớp: 03

Phần đọc và triển khai

Phần: **1.3. Tạo quy trình với system()**

Trả lời các câu hỏi dưới đây:

a. Giải thích cú pháp của hàm **system()**?

b. Đặc điểm của các quy trình được tạo bởi **system()**?

- Quan hệ với quá trình ban đầu? PID? Mã nguồn? Làm thế nào chúng ta có thể xác định / phân công công việc cho họ?

1. Tạo tiến trình

Bài tập trên lớp: 04

Vẽ bảng và so sánh những điểm giống và khác nhau giữa việc tạo các quy trình bằng cách sử dụng fork(), exec() và system()?

Dựa trên sự so sánh này, hãy giải thích trong trường hợp nào mỗi chức năng nên được sử dụng?

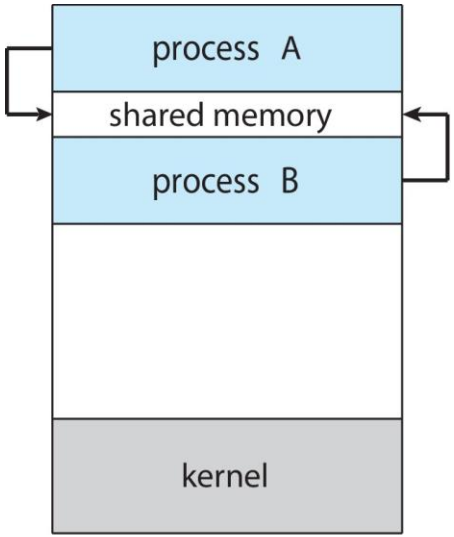
GIAO TIẾP GIỮA CÁC TIẾN TRÌNH

02.

2. Giao tiếp giữa các tiến trình

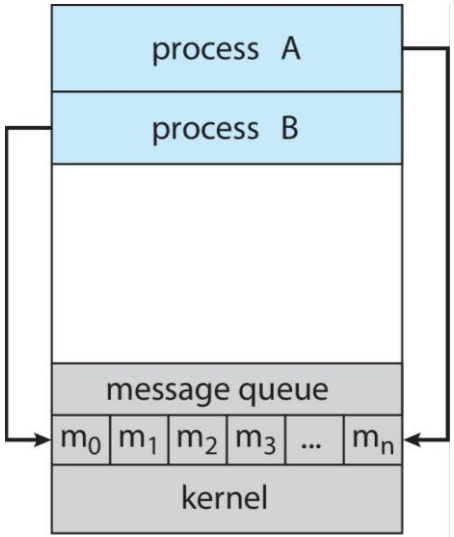
Có 02 cơ chế:

Bộ nhớ dùng chung



(a)

Truyền tin nhắn



(b)

2.1. Truyền tin nhắn

- Trong các hệ điều hành tương tự Unix, **pipes** được sử dụng cho giao tiếp giữa các quá trình (IPC), cho phép một quá trình gửi dữ liệu đến một quy trình khác.

2.1. Truyền tin nhắn

Quy trình:

1 Tạo một Pipe

Sử dụng lệnh hệ thống pipe() để tạo đường ống. Lệnh gọi này trả về hai mô tả tệp: một để đọc và một để viết

System call pipe()

- **pipefd[0]**: Mô tả tệp để đọc từ pipe
- **pipefd[1]**: Mô tả tệp để ghi vào pipe

2.1. Truyền tin nhắn

Quy trình:

2 **Fork một quy trình con**

Dùng lệnh hệ thống pipe() để tạo đường ống. Lệnh gọi này trả về hai mô tả tệp: một để đọc và một để viết

3 **Đóng các kết thúc không cần thiết**

Ví dụ: Trong quá trình cha, đóng đầu đọc của đường ống. Trong quá trình con, đóng đầu viết của đường ống

2.1. Truyền tin nhắn

Quy trình :

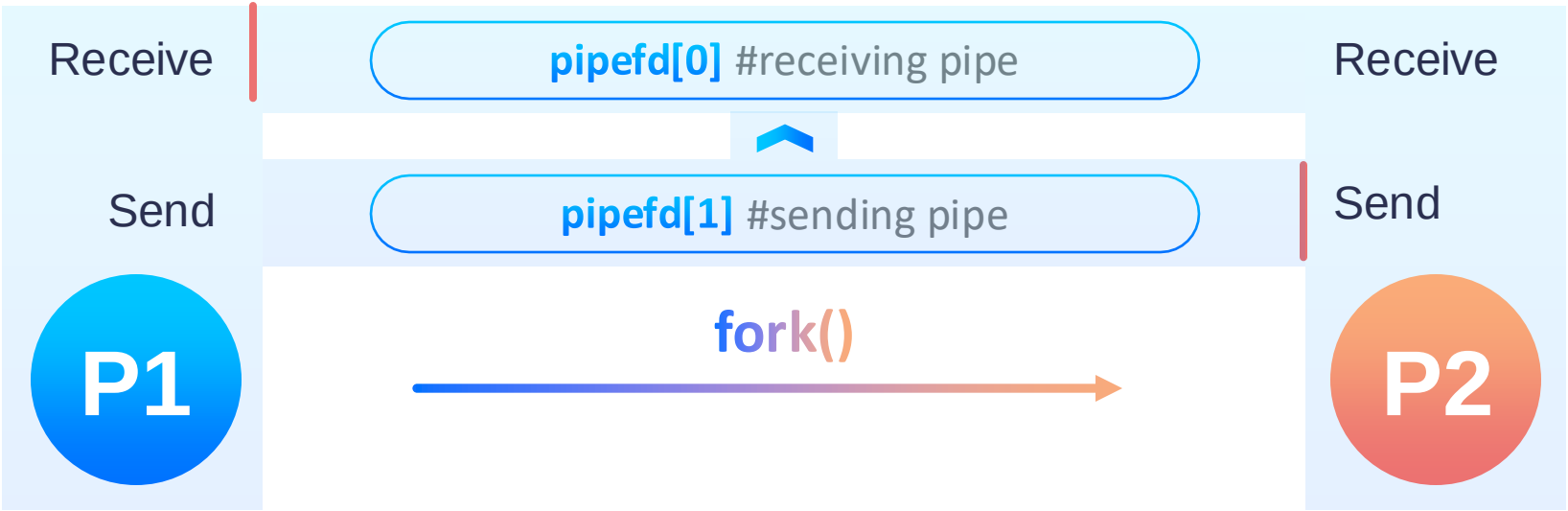
4 **Viết và đọc**

Ví dụ: Tiến trình cha ghi dữ liệu vào pipe và tiến trình con đọc từ pipe

5 **Đóng đường ống**

Sau khi giao tiếp xong, đóng đường ống trong cả hai quy trình

2.2. Ví dụ sử dụng pipe



Phần: 2. Giao tiếp giữa các quá trình: Truyền thông điệp

BÀI TẬP

Phần này sẽ được cập nhật sau buổi học trên lớp

03.

3. Bài tập

Bài tập : 01

Viết chương trình time.cpp Điều đó đo thời gian thực hiện của một lệnh shell. Chương trình sẽ được thực thi với cú pháp./time <command> nơi mà <command> là lệnh shell mà bạn muốn đo thời gian thực hiện. Ví dụ:

```
$ ./time ls
time.c
time
Execution time: 0.25422
```

3. Exercises

Exercise: 02

Thiết kế một chương trình sao chép tệp có tên filecopy.cpp sử dụng pipe thông thường. Chương trình này sẽ được thông qua hai tham số: tên của tệp sẽ được sao chép và tên của tệp đích. Sau đó, chương trình sẽ tạo một đường ống thông thường và ghi nội dung của tệp sẽ được sao chép vào đường ống. Tiến trình con sẽ đọc tập tin này từ pipe và ghi nó vào file đích. Ví dụ: nếu chúng ta gọi chương trình như sau:./filecopy input.txt copy.txt

Tệp input.txt sẽ được ghi vào pipeline. Tiến trình con sẽ đọc nội dung của tập tin này và ghi nó vào tập tin đích copy.txt.

3. Bài tập

Bonus

Giả thuyết Collatz liên quan đến những gì xảy ra khi chúng ta lấy bất kỳ số nguyên dương n nào và áp dụng thuật toán sau:

$$n = \begin{cases} n/2, & \text{if n is even} \\ 3 \times n + 1, & \text{if n is odd} \end{cases}$$

Phỏng đoán nói rằng khi thuật toán này được áp dụng liên tục, tất cả các số nguyên dương cuối cùng sẽ đạt đến 1. Ví dụ: nếu n = 35, dãy là *35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1*

3. Bài tập

Bonus (cont.)

Viết chương trình C bằng cách sử dụng hệ thống lệnh **fork()** tạo ra trình tự này trong quá trình con. Số bắt đầu sẽ được cung cấp từ dòng lệnh. Ví dụ: nếu **8** được truyền dưới dạng tham số trên dòng lệnh, tiến trình con sẽ xuất ra **8, 4, 2, 1**. Bởi vì các quy trình cha và con có các bản sao dữ liệu riêng, nên trẻ sẽ cần phải xuất trình tự. Yêu cầu phụ huynh gọi **wait()** Gọi để chờ quá trình con hoàn tất trước khi thoát khỏi chương trình. Thực hiện kiểm tra lỗi cần thiết để đảm bảo rằng một số nguyên dương được truyền trên dòng lệnh.