|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **MSSV** | **Lớp** |
| Lại Quan Thiên | 22521385 | IT007.O21.1 |
| Đặng Đức Tài | 22521270 |
| Mai Nguyễn Nam Phương | 22521164 |
| Phùng Trần Thế Nam | 21522366 |

HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 3

**CHECKLIST**

**3.5. BÀI TẬP THỰC HÀNH**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **BT 1** | **BT 2** | **BT 3** | **BT 4** |
| **Trình bày cách làm** |  |  |  |  |
| **Chụp hình minh chứng** |  |  |  |  |
| **Giải thích kết quả** |  |  |  |  |

**3.6. BÀI TẬP ÔN TẬP**

|  |  |
| --- | --- |
|  | **BT 1** |
| **Trình bày cách làm** |  |
| **Chụp hình minh chứng** |  |
| **Giải thích kết quả** |  |

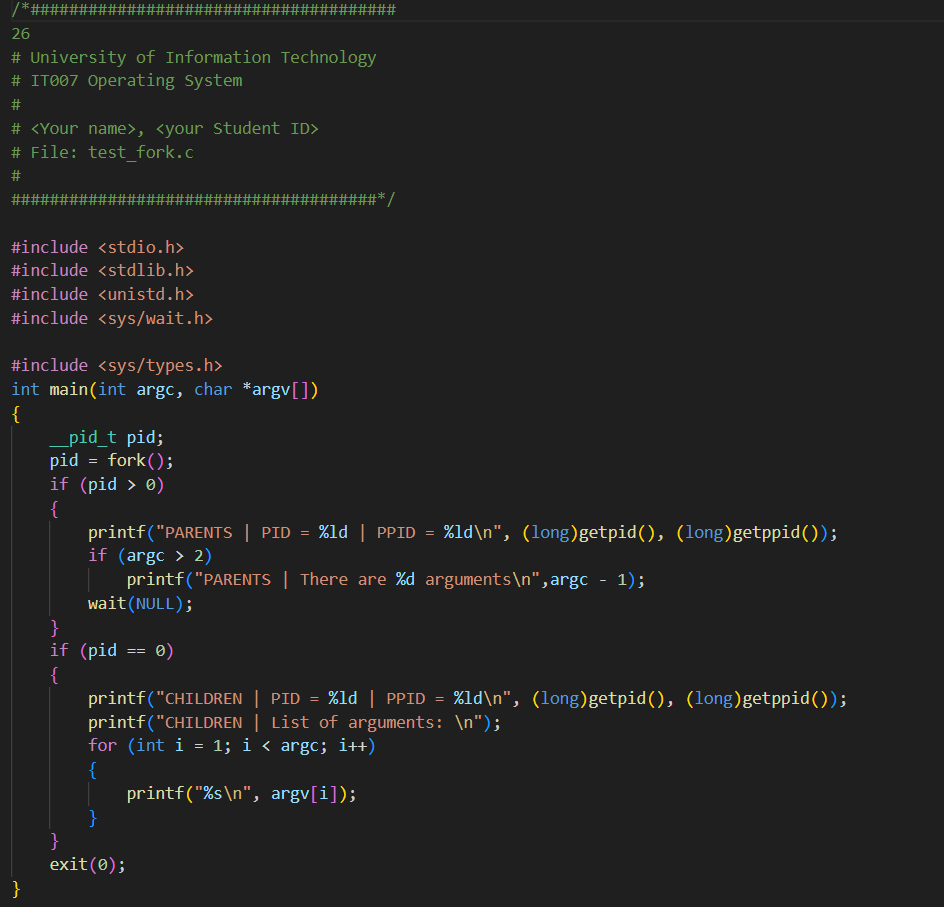
**Tự chấm điểm:** 10/10

*\*Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:* ***<MSSV>\_LAB3.pdf***

**3.7. BÀI TẬP THỰC HÀNH**

# Thực hiện Ví dụ 3-1, Ví dụ 3-2, Ví dụ 3-3, Ví dụ 3-4 giải thích code và kết quả nhận được?

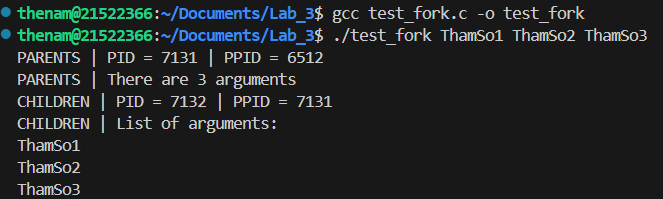
**- Ví dụ 3-1:**



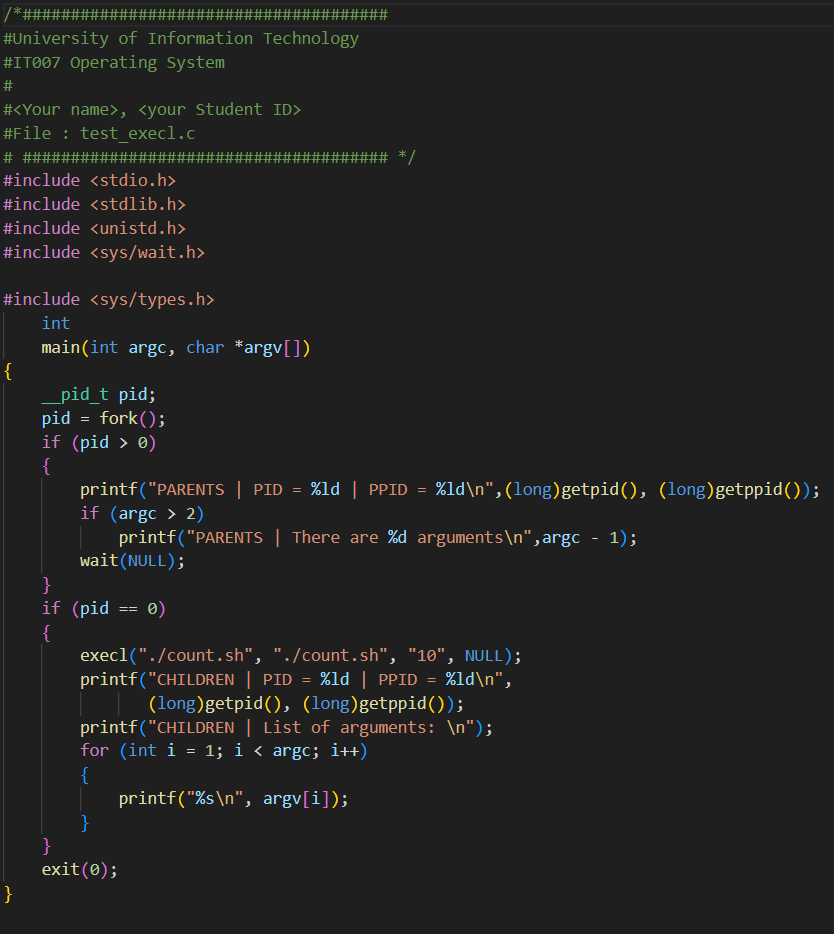
**+ Giải thích source code:**

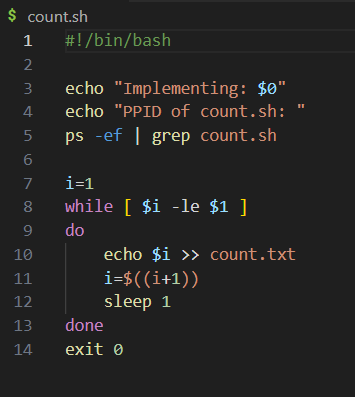
* Hàm fork() tạo một bản sao của tiến trình gọi nó. Trong trường hợp này, nó tạo ra một tiến trình con.
* Xử lý trong tiến trình cha: Nếu pid > 0, đây là tiến trình cha. Chương trình in ra PID và PPID của tiến trình cha. Nếu số lượng tham số truyền vào lớn hơn 2, chương trình in ra số lượng tham số đó. Sau đó, tiến trình cha chờ tiến trình con kết thúc bằng hàm wait(NULL).
* Xử lý trong tiến trình con: Nếu pid == 0, đây là tiến trình con. Chương trình in ra PID và PPID của tiến trình con, sau đó in ra danh sách các tham số truyền vào.

**+ Kết quả:**



**- Ví dụ: 3-2:**

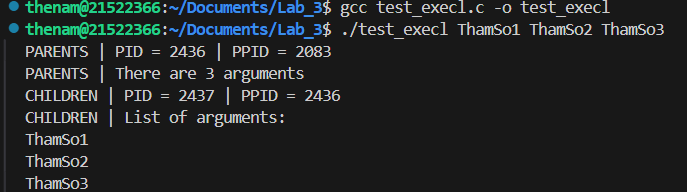




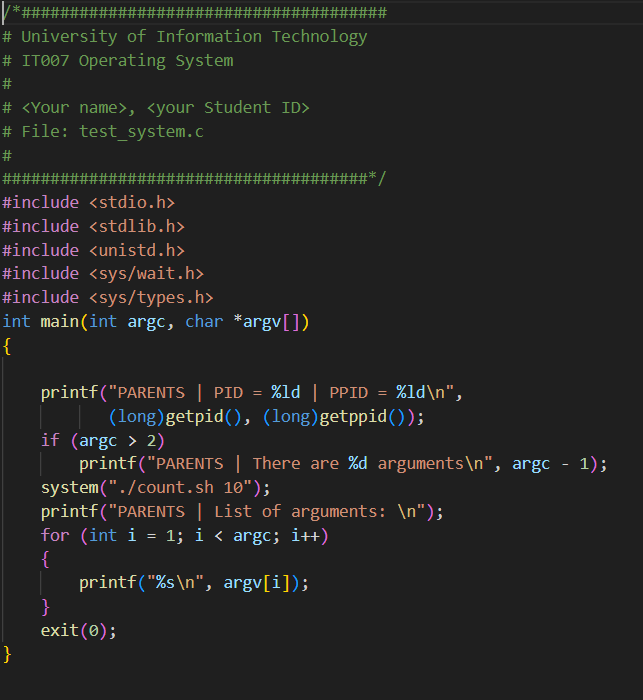
**+ Giải thích source code:**

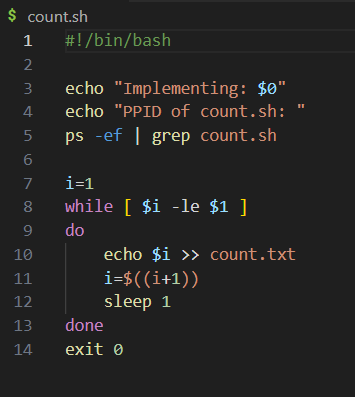
* Tạo tiến trình con với fork(): Sử dụng hàm fork() để tạo một tiến trình con. Giá trị trả về của fork() là 0 trong tiến trình con và một số khác trong tiến trình cha.
* Xử lý trong tiến trình cha: Nếu pid lớn hơn 0, tức là đang ở trong tiến trình cha, in ra thông tin về PID và PPID của tiến trình cha. Nếu có hơn 2 tham số dòng lệnh, in ra số lượng tham số.
* Xử lý trong tiến trình con: Nếu pid bằng 0, tức là đang ở trong tiến trình con, sử dụng hàm execl() để thực thi tập lệnh count.sh với tham số là "10". In ra thông tin về PID và PPID của tiến trình con. In ra các tham số dòng lệnh truyền vào.

**+ Kết quả:**



**- Ví dụ: 3-3:**

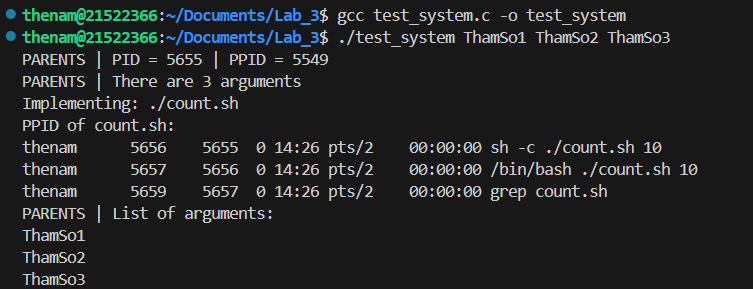




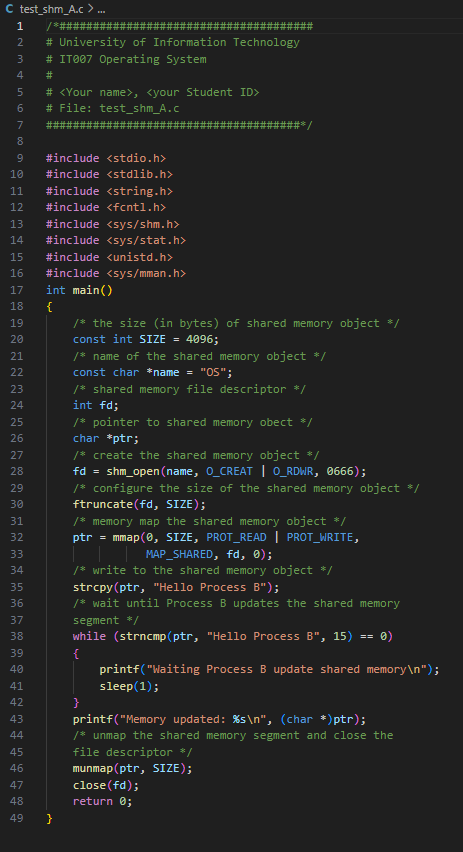
**+ Giải thích source code:**

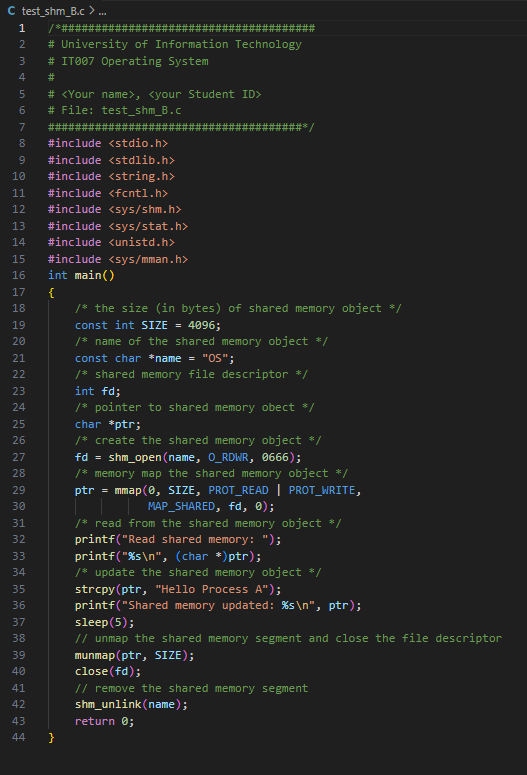
* Xử lý trong tiến trình cha: In ra thông tin về PID và PPID của tiến trình cha. Nếu có hơn 2 tham số dòng lệnh, in ra số lượng tham số.
* Thực thi lệnh với hàm system(): Sử dụng hàm system() để thực thi lệnh ./count.sh 10. Điều này sẽ gọi script count.sh với tham số là 10.

**+ Kết quả:**



**- Ví dụ: 3-4:**





**+ Giải thích source code:**

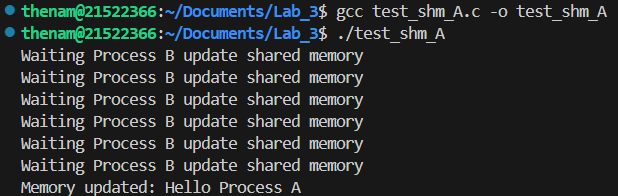
**Process A (test\_shm\_A.c):**

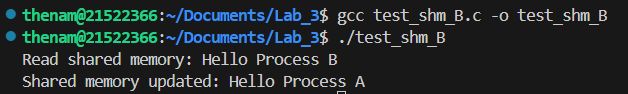
* Mô tả: Quy trình A tạo một đối tượng shared memory, ghi dữ liệu vào nó và sau đó chờ quy trình B cập nhật dữ liệu trong đối tượng shared memory. Khi quy trình B đã cập nhật xong, quy trình A đọc và in ra dữ liệu đã được cập nhật.
* Bước thực thi:
* Tạo một đối tượng shared memory bằng hàm shm\_open().
* Cấu hình kích thước của đối tượng shared memory bằng ftruncate().
* Ánh xạ đối tượng shared memory vào không gian địa chỉ của quy trình bằng mmap().
* Ghi dữ liệu vào đối tượng shared memory bằng cách sao chép chuỗi "Hello Process B" vào con trỏ ptr.
* Chờ đợi cho đến khi quy trình B cập nhật dữ liệu trong shared memory thông qua vòng lặp while.
* Khi dữ liệu đã được cập nhật, đọc và in ra dữ liệu đã được cập nhật.
* Hủy ánh xạ và đóng file descriptor bằng munmap() và close().

**Process B (test\_shm\_B.c):**

* Mô tả: Quy trình B mở đối tượng shared memory được tạo bởi quy trình A, đọc dữ liệu từ đối tượng shared memory, sau đó cập nhật dữ liệu trong nó. Sau đó, quy trình B chờ một thời gian trước khi hủy bỏ đối tượng shared memory.
* Bước thực thi:
* Mở đối tượng shared memory bằng hàm shm\_open() với quyền đọc và ghi.
* Ánh xạ đối tượng shared memory vào không gian địa chỉ của quy trình bằng mmap().
* Đọc dữ liệu từ đối tượng shared memory và in ra màn hình.
* Cập nhật dữ liệu trong đối tượng shared memory bằng cách sao chép chuỗi "Hello Process A" vào con trỏ ptr.
* Chờ một khoảng thời gian bằng hàm sleep().
* Hủy ánh xạ và đóng file descriptor bằng munmap() và close().
* Xóa đối tượng shared memory bằng hàm shm\_unlink().
* Một khi quy trình B đã cập nhật xong dữ liệu trong shared memory, quy trình A sẽ tiếp tục thực hiện và in ra dữ liệu đã được cập nhật. Điều này minh họa cơ chế truyền thông qua shared memory giữa các quy trình trong hệ điều hành.

**+ Kết quả:**





# Viết chương trình time.c thực hiện đo thời gian thực thi của một lệnh shell. Chương trình sẽ được chạy với cú pháp "./time <command>" với <command> là lệnh shell muốn đo thời gian thực thi.

Ví dụ:

$ ./time ls

time.c

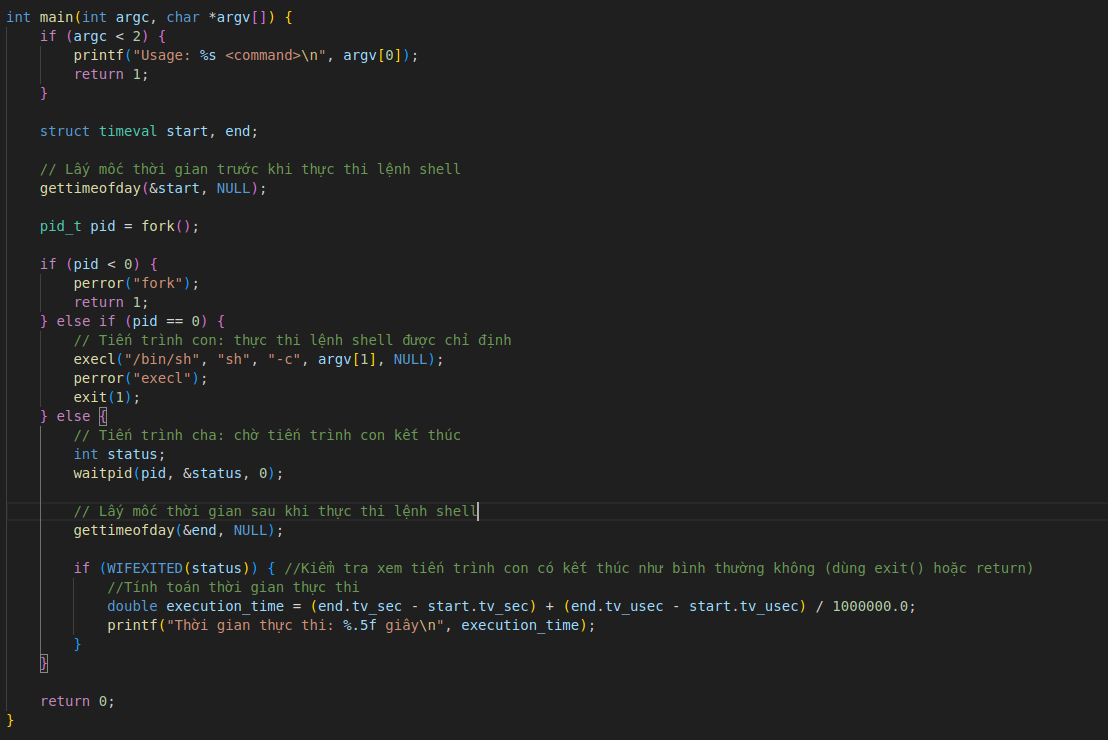
time

Thời gian thực thi: 0.25422

**Gợi ý: Tiến trình cha gọi hàm fork() tạo ra tiến trình con rồi wait(). Tiến trình con gọi hàm gettimeofday() để lấy mốc thời gian trước khi thực thi lệnh shell, sau đó sử dụng hàm execl() để thực thi lệnh. Sau khi tiến trình con kết thúc, tiến trình cha tiếp tục gọi hàm gettimeifday() một lần nữa để lấy mốc thời gian sau khi thực thi lệnh shell và tính toán.**

**Trả lời:**

**- Source code:** (giải thích trong comments)



**- Kết quả:**

A screenshot of a computer

Description automatically generated

**- Giải thích:**

* Trước khi thực thi lệnh shell, chương trình sẽ lấy mốc thời gian bắt đầu bằng cách sử dụng hàm gettimeofday().
* Tiến trình cha sẽ tạo một tiến trình con bằng cách sử dụng hàm fork().
* Trong tiến trình con, lệnh shell được thực thi bằng hàm execl() và tham số dòng lệnh được truyền vào.
* Tiến trình cha sẽ đợi tiến trình con kết thúc bằng hàm waitpid().
* Sau khi tiến trình con kết thúc, tiến trình cha lấy mốc thời gian kết thúc bằng cách sử dụng lại hàm gettimeofday().
* Sau đó, chương trình tính toán thời gian thực thi bằng cách lấy hiệu của thời gian kết thúc và thời gian bắt đầu.
* Kết quả thời gian thực thi được in ra màn hình với định dạng số thập phân có 5 chữ số sau dấu thập phân.

# Viết một chương trình làm bốn công việc sau theo thứ tự:

# In ra dòng chữ: “Welcome to IT007, I am <your\_Student\_ID>!”

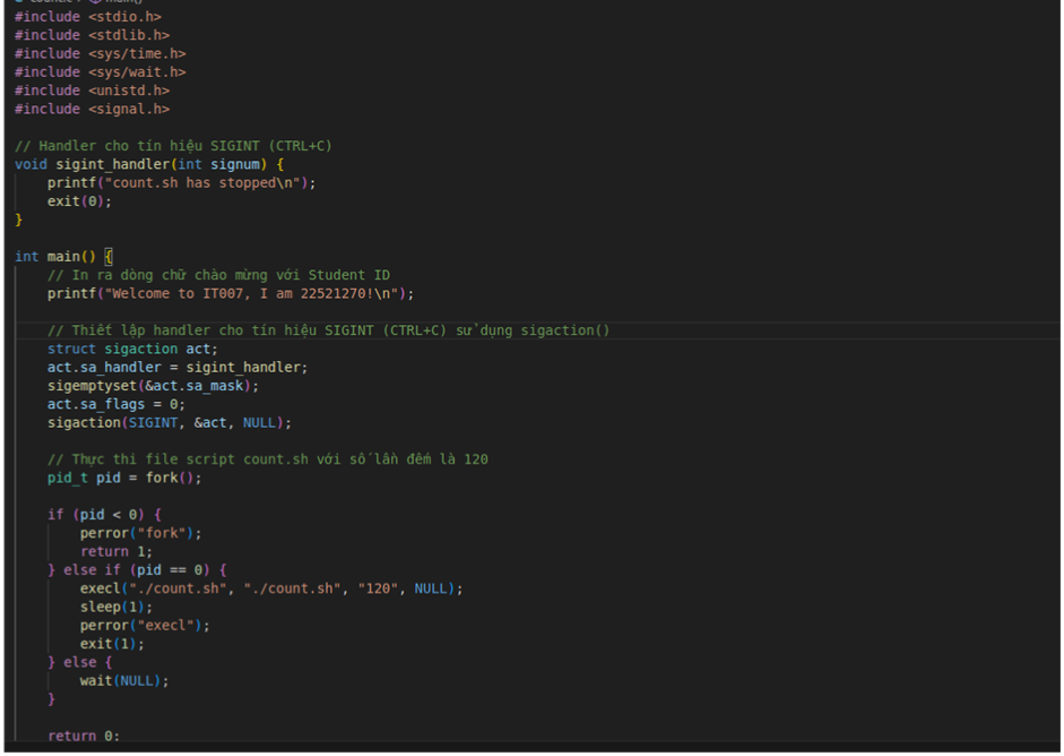
# Thực thi file script count.sh với số lần đếm là 120

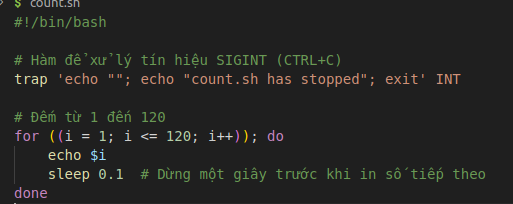
# Trước khi count.sh đếm đến 120, bấm CTRL+C để dừng tiến trình này

# Khi người dùng nhấn CTRL+C thì in ra dòng chữ: “count.sh has stoppped”s

**Trả lời:**

**- Source code:** (giải thích trong comments)





**- Kết quả:**

A screenshot of a computer

Description automatically generated

**- Giải thích:**

* In ra dòng chữ chào mừng với Student ID.
* Thiết lập một handler cho tín hiệu SIGINT (CTRL+C) bằng cách sử dụng sigaction().
* Thực thi file script count.sh với số lần đếm là 120 bằng cách sử dụng hàm execl().
* Nếu không có lỗi xảy ra trong quá trình thực thi file script, tiến trình con sẽ kết thúc và tiến trình cha sẽ đợi cho đến khi tiến trình con kết thúc.
* Nếu có lỗi xảy ra trong quá trình thực thi file script, tiến trình con sẽ in ra thông báo lỗi và thoát với mã lỗi 1.
* Khi tiến trình cha nhận được tín hiệu SIGINT (CTRL+C), nó sẽ kích hoạt handler sigint\_handler(), in ra thông báo "count.sh has stopped" và thoát chương trình.
* Nếu tập lệnh count.sh thực hiện đúng, chương trình sẽ kết thúc và không cần phải in ra thông báo lỗi từ perror("execl").

# Viết chương trình mô phỏng bài toán Producer - Consumer như sau:

# Sử dụng kỹ thuật shared-memory để tạo một bounded-buffer có độ lớn là 10 bytes.

# Tiến trình cha đóng vai trò là Producer, tạo một số ngẫu nhiên trong khoảng [10, 20] và ghi dữ liệu vào buffer

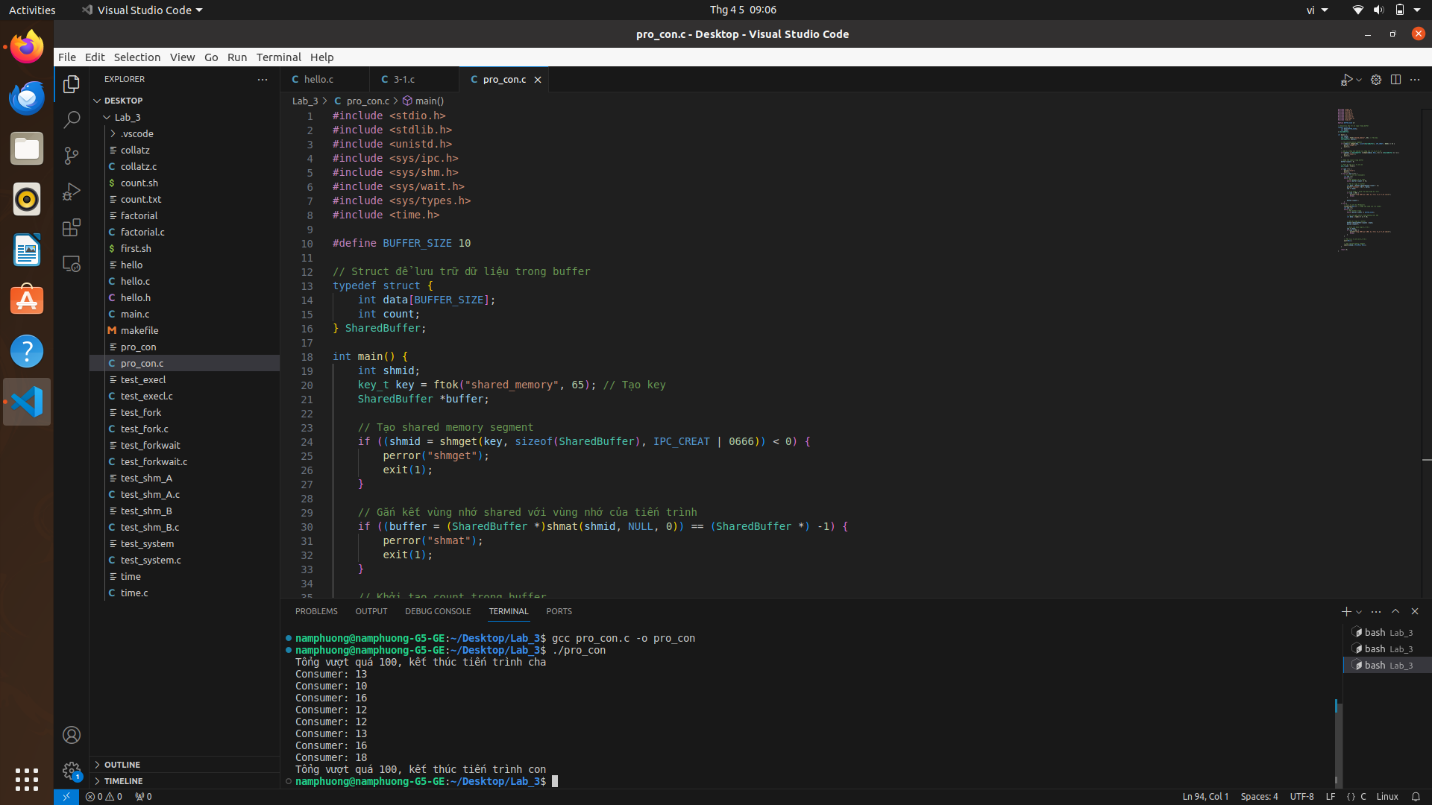
# Tiến trình con đóng vai trò là Consumer đọc dữ liệu từ buffer, in ra màn hình và tính tổng

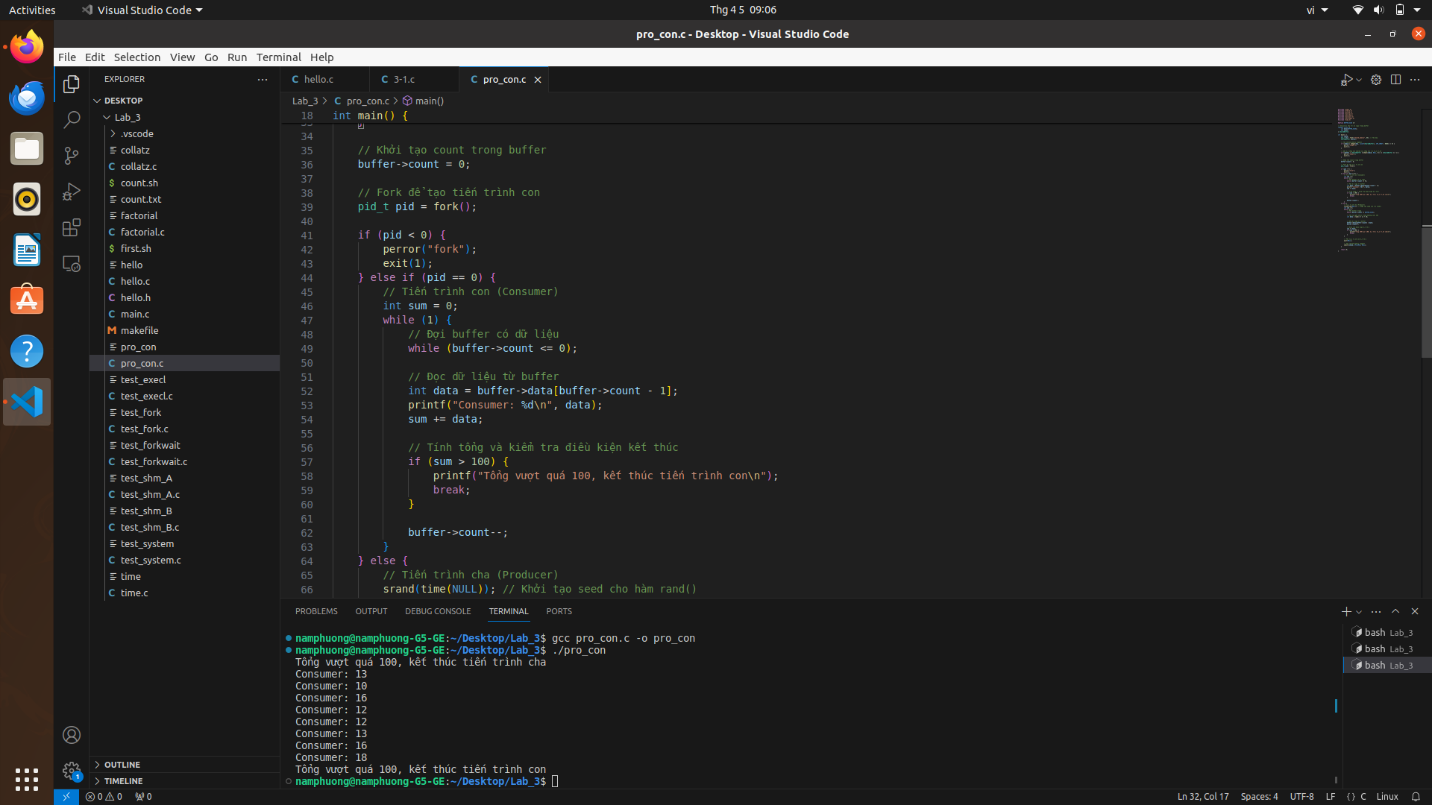
# Khi tổng lớn hơn 100 thì cả 2 dừng lại

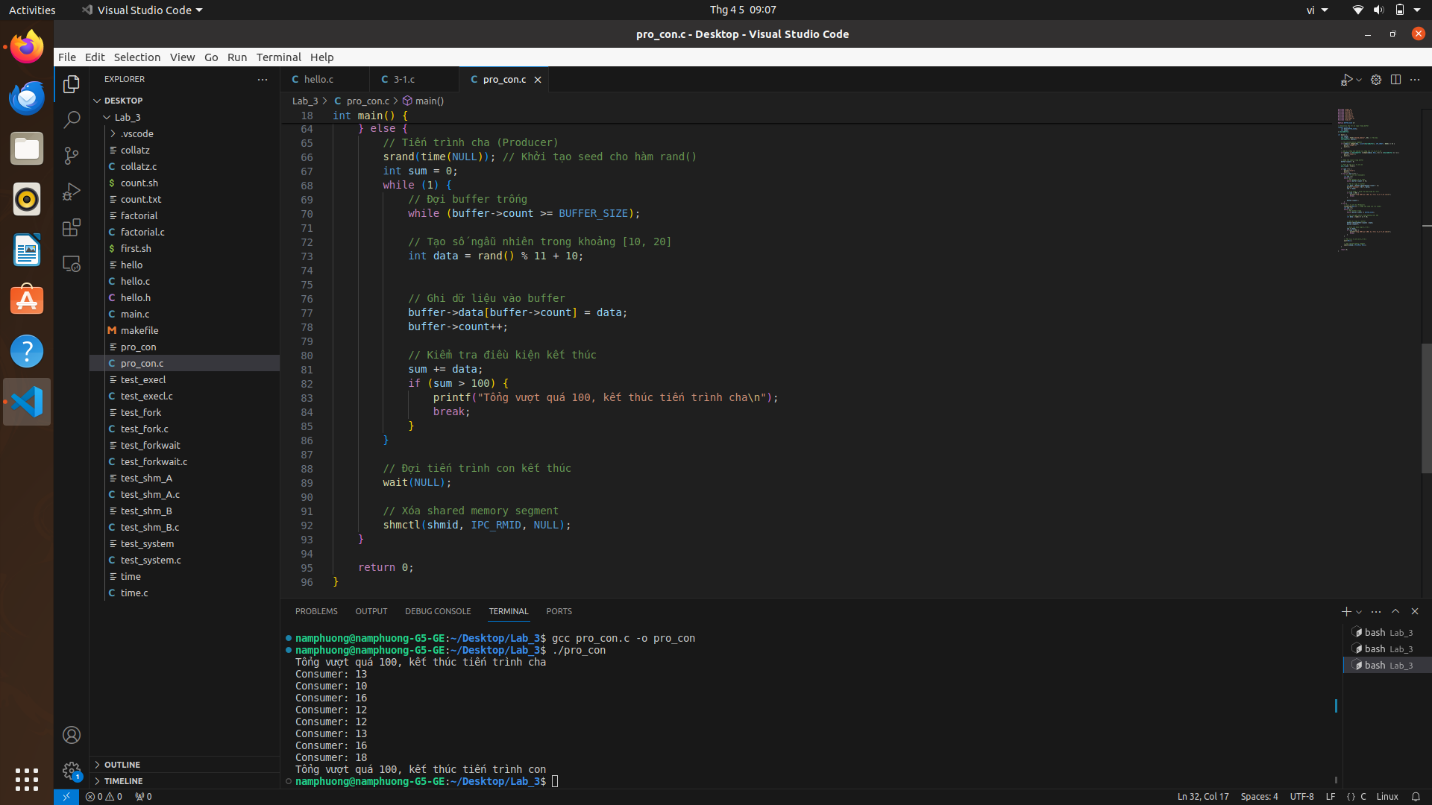
**Trả lời:**

**- Trình bày cách làm:** đã comment trong code

**- Chụp hình minh chứng:**







**- Kết quả:**

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phông chữ

Mô tả được tạo tự động

**- Giải thích kết quả:**

+ Chương trình trên tạo hai tiến trình, một tiến trình cha là Producer và một tiến trình con là Consumer,sử dụng shared memory để trao đổi dữ liệu giữa chúng.

+ Tiến trình cha (Producer): Tiến trình cha sinh ra các số ngẫu nhiên trong khoảng từ 10 đến 20 và ghi chúng vào buffer được chia sẻ. Sau đó, tiến trình cha kiểm tra tổng của các số đã ghi vào buffer, nếu tổng vượt quá 100, tiến trình cha kết thúc.

+ Tiến trình con (Consumer): Tiến trình con đợi cho đến khi buffer có ít nhất một số được ghi vào. Sau đó, nó lấy số cuối cùng từ buffer, in ra màn hình và cộng vào tổng. Nếu tổng vượt quá 100, tiến trình con kết thúc.

**- Các bước thực hiện:**

+ Tiến trình cha (Producer) sẽ sinh ra các số ngẫu nhiên và in chúng ra màn hình.

+ Tiến trình con (Consumer) sẽ tiếp tục in ra màn hình các số mà tiến trình cha ghi vào buffer.

+ Khi tổng của các số vượt quá 100, cả hai tiến trình đều sẽ kết thúc.

+ Lưu ý: Đối với các chương trình sử dụng shared memory như vậy, kết quả cuối cùng có thể thay đổi mỗi lần chạy do sự ngẫu nhiên trong sinh số ngẫu nhiên và thời điểm các tiến trình kết thúc.

**3.8. BÀI TẬP ÔN TẬP**

# Phỏng đoán Collatz xem xét chuyện gì sẽ xảy ra nếu ta lấy một số nguyên dương bất kỳ và áp dụng theo thuật toán sau đây: Phỏng đoán phát biểu rằng khi thuật toán này được áp dụng liên tục, tất cả số nguyên dương đều sẽ tiến đến 1. Ví dụ, với n = 35, ta sẽ có chuỗi kết quả như sau:

35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Viết chương trình C sử dụng hàm fork() để tạo ra chuỗi này trong tiến trình con. Số bắt đầu sẽ được truyền từ dòng lệnh. Ví dụ lệnh thực thi ./collatz 8 sẽ chạy thuật toán trên n = 8 và chuỗi kết quả sẽ ra là 8, 4, 2, 1. Khi thực hiện, tiến trình cha và tiến trình con chia sẻ một buffer, sử dụng phương pháp bộ nhớ chia sẻ, hãy tính toán chuỗi trên tiến trình con, ghi kết quả vào buffer và dùng tiến trình cha để in kết quả ra màn hình. Lưu ý, hãy nhớ thực hiện các thao tác để kiểm tra input là số nguyên dương.

**Trả lời:**

- Chương trình được gọi với một đối số là số nguyên dương. Nếu không, màn hìnhhình sẽ hiển thị một thông báo lỗi và thoát.

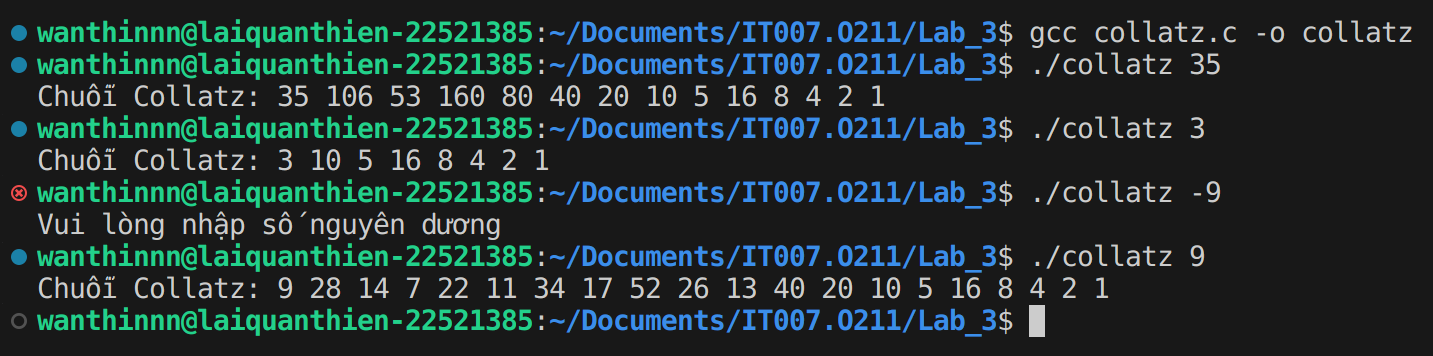
- Tiến trình cha tạo một shared memory segment và gắn kết nó vào vùng nhớ của tiến trình. Sau đó, nó gọi fork() để tạo một tiến trình con.

- Tiến trình con tính toán chuỗi Collatz cho số nguyên được cung cấp và lưu trữ nó trong shared memory segment.

- Tiến trình cha đợi cho tiến trình con kết thúc, sau đó in ra chuỗi kết quả từ shared memory segment.

- Cuối cùng, chúng ta xóa shared memory segment.

**- Minh họa:**



**- Source code:**

