|  |
| --- |
| Họ và tên: Đỗ Phương Duy  Mã số sinh viên: 23520362  Lớp: KHTN2023 |

HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 3

**CHECKLIST (Đánh dấu x khi hoàn thành)**

**Lưu ý mỗi câu phải làm đủ 3 yêu cầu**

1. **Bài tập thực hành**

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

1. **Bài tập ôn tập**

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

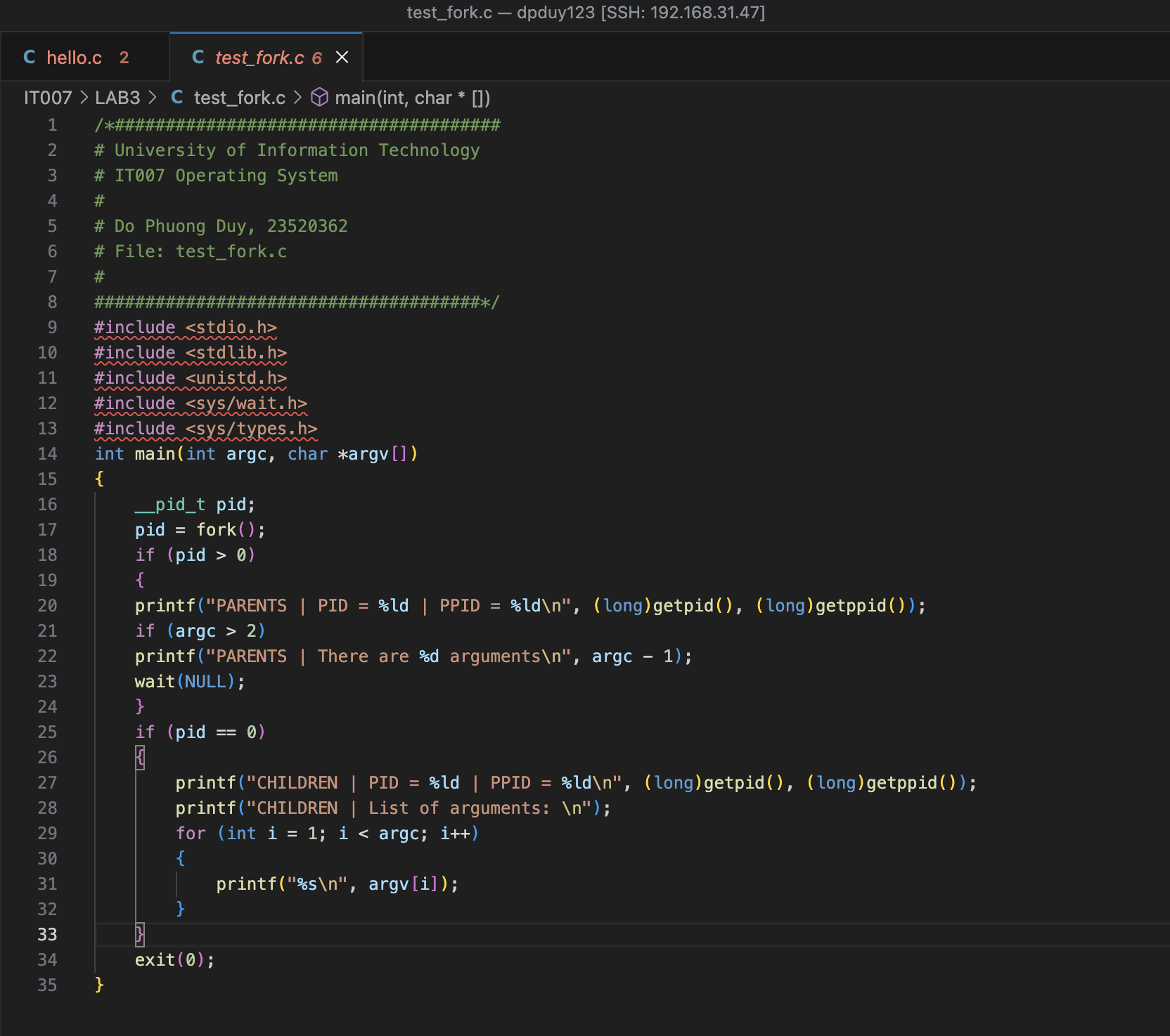
**Tự chấm điểm:**  10

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

1. **Bài tập thực hành:**

# 1.

**A. Ví dụ 3-1:**

Code: 

Giải thích code:

Hàm main nhận số lượng tham số argc và mảng các tham số (agrv[])

\_\_pid\_t pid và pid = fork() là lệnh tạo tiến trình con.

Fork() tạo ra tiến trình con, gọi X là giá trị hàm này trả về. Nếu:

* X > 0 là tiến trình cha (PID của tiến trình con)
* X = 0 là tiến trình con
* X < 0 là đã có lỗi xảy ra

Nếu pid lớn hơn 0, nghĩa là đoạn mã này đang chạy trong tiến trình cha.

Nó in ra PID (Process ID) và PPID (Parent Process ID) của tiến trình cha.

Nếu có nhiều hơn 2 tham số (tính cả tên chương trình), nó sẽ in ra số lượng tham số.

Cuối cùng, tiến trình cha gọi wait(NULL) để chờ tiến trình con hoàn thành trước khi tiếp tục.

Nếu pid bằng 0, đoạn mã này đang chạy trong tiến trình con.

Chương trình in ra PID và PPID của tiến trình con.

Sau đó, in ra danh sách các tham số được truyền vào từ argv, bắt đầu từ argv[1] (bỏ qua tên chương trình ở argv[0]).

Biên dịch:



Giải thích: biên dịch code trên bằng lệnh ./test\_fork 128 64 32 với 3 tham số là 128 64 32.

PID và PPID của tiến trình cha lần lượt là 6777 và 4163

PID và PPID của tiến trình con lần lượt là 6778 và 6777

**B. Ví dụ 3-2:**

**execl("./count.sh", "./count.sh", "10", NULL);: Thực thi tập lệnh shell count.sh với tham số "10". Nếu execl thành công, mã phía dưới sẽ không được thực thi.**

Code:

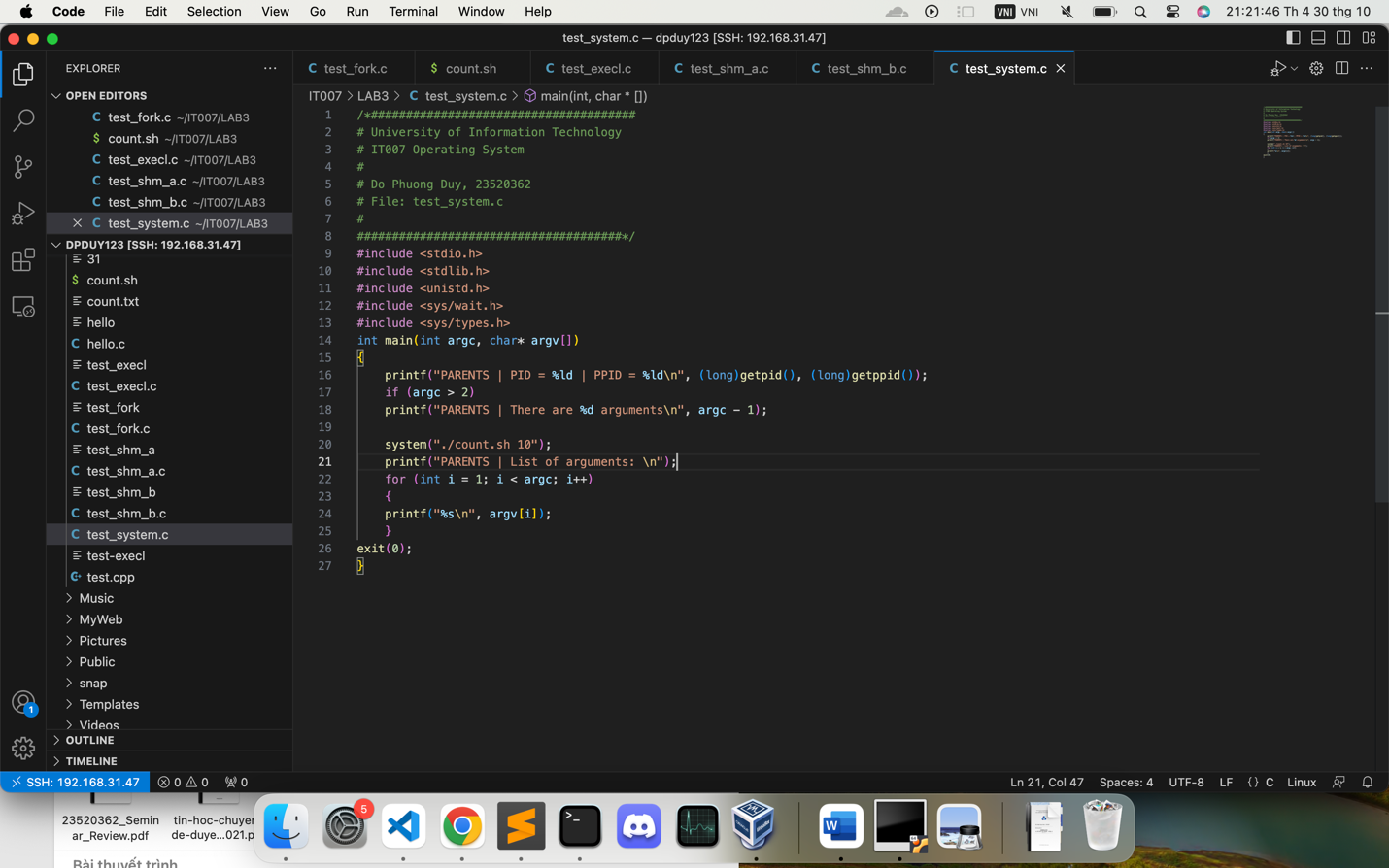


Biên dịch:

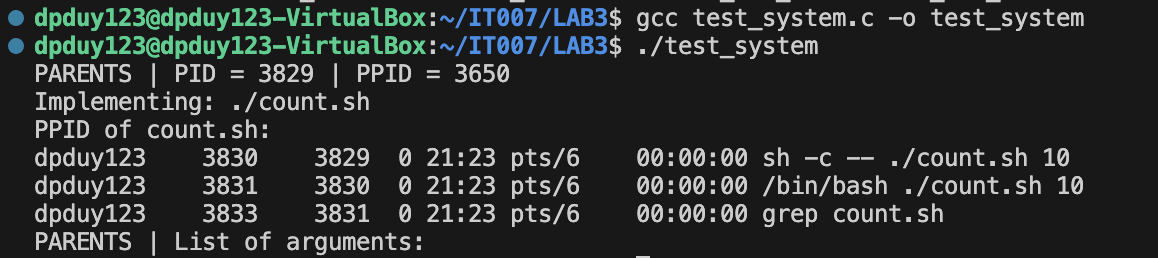


Chương trình này minh họa việc tạo và xử lý tiến trình con trong C. Tiến trình cha in ra thông tin về nó và chờ tiến trình con. Tiến trình con thực thi một tập lệnh shell với tham số, và nếu việc thực thi thành công, nó sẽ không thực hiện các lệnh phía sau execl(). Chương trình sử dụng fork() để phân chia tiến trình và execl() để thay thế tiến trình con bằng một chương trình khác.

**C. Ví dụ 3-3:**

Code:

Biên dịch:



Giải thích:

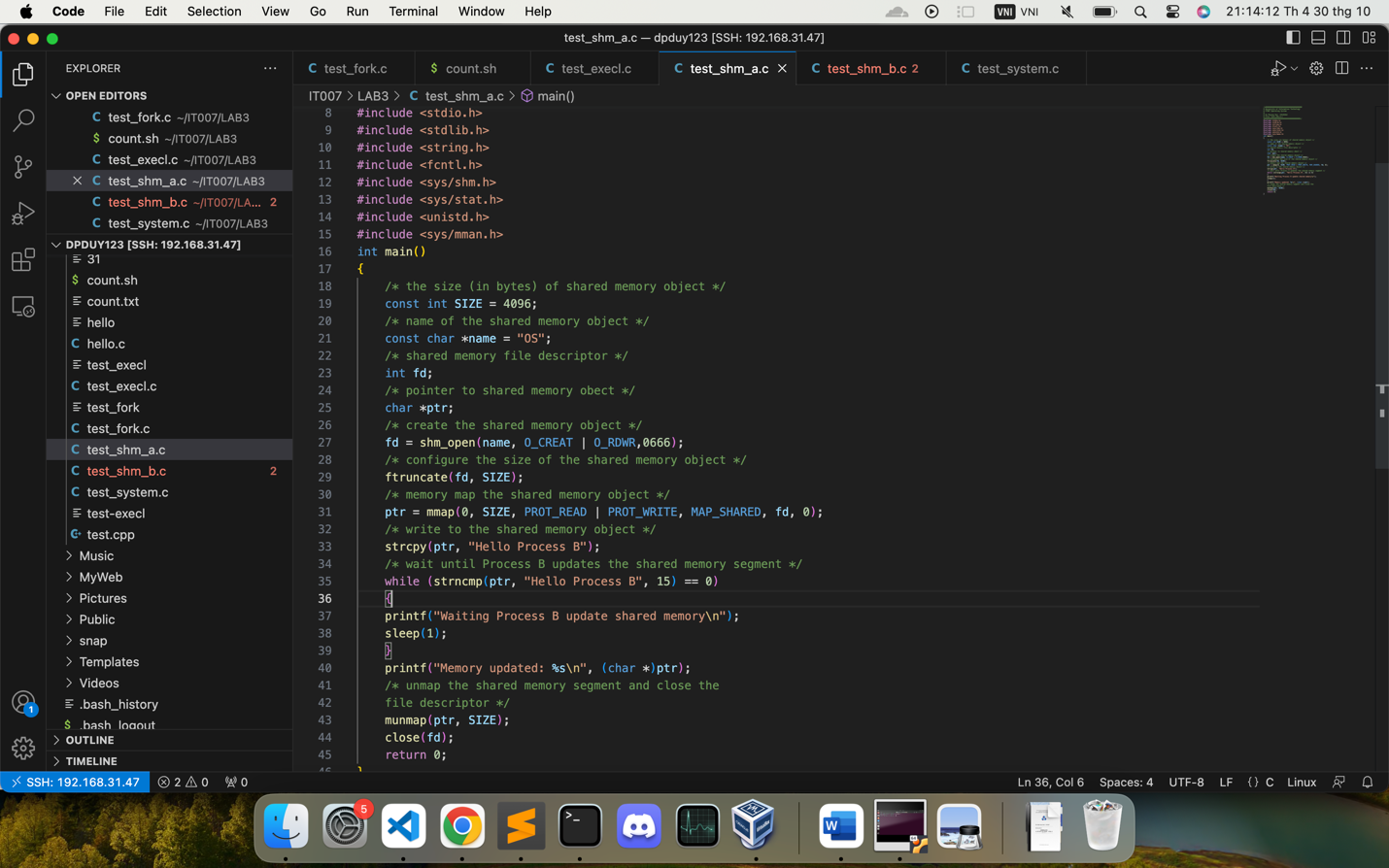
Chương trình này minh họa cách sử dụng hàm system() để thực thi một lệnh shell từ trong một chương trình C. Nó in ra thông tin về PID và PPID, kiểm tra số lượng tham số, và sau đó thực thi tập lệnh count.sh với tham số "10". Cuối cùng, nó in ra danh sách các tham số đã truyền vào.

**D. Ví dụ 3-4:**

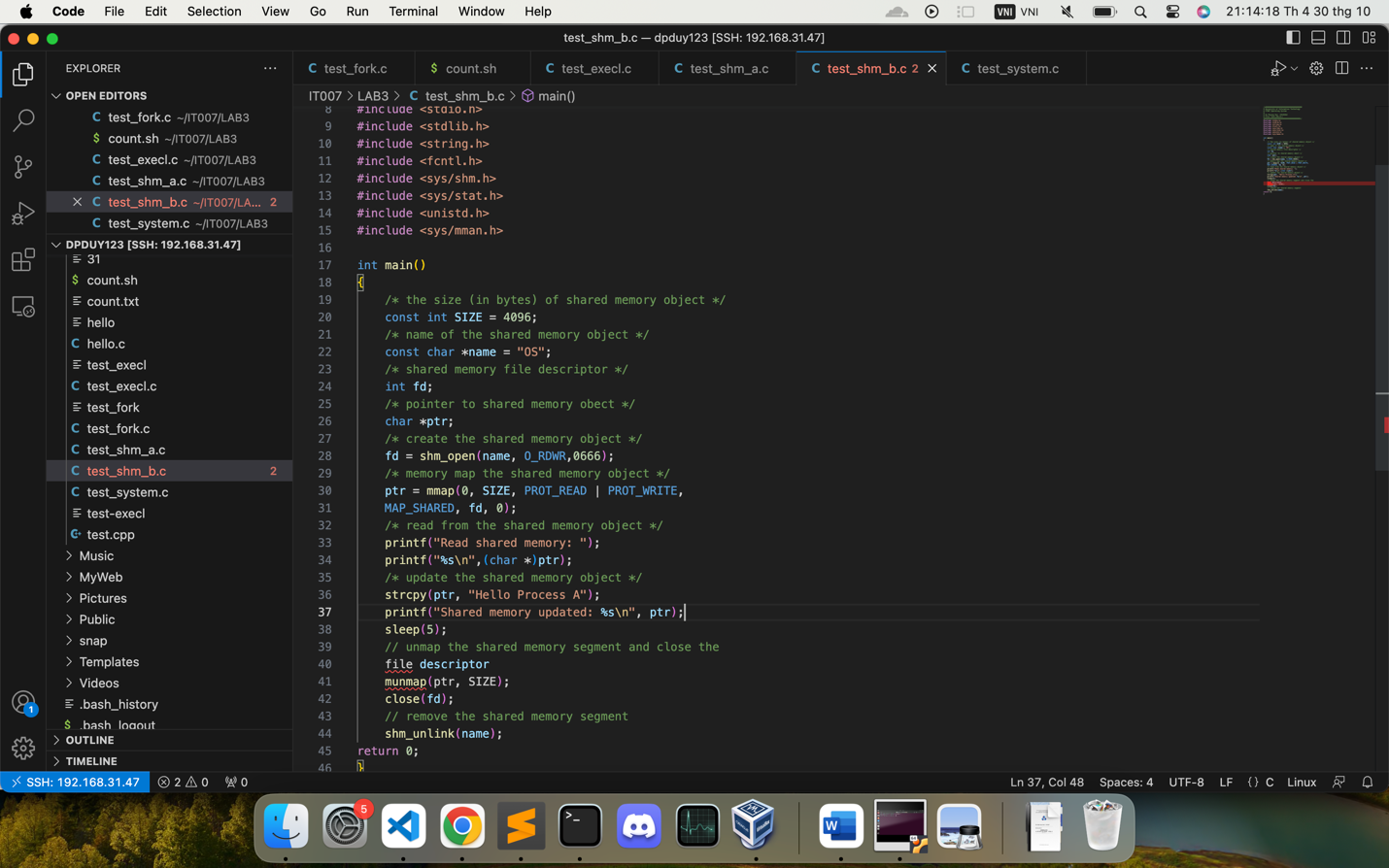
Process A khởi tạo bộ nhớ chia sẻ, ghi vào bộ nhớ "Hello Process B", sau đó chờ bộ nhớ được cập nhật bởi Process B.

Proces B truy cập bộ nhớ chia sẻ, đọc dữ liệu do Process A ghi vào, sau đó cập nhật bộ nhớ với chuỗi "Hello Process A"

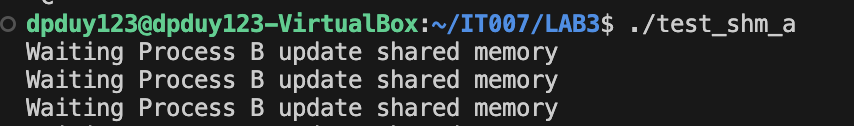
Code tiến trình A:



Code tiến trình B:



Chạy process A:



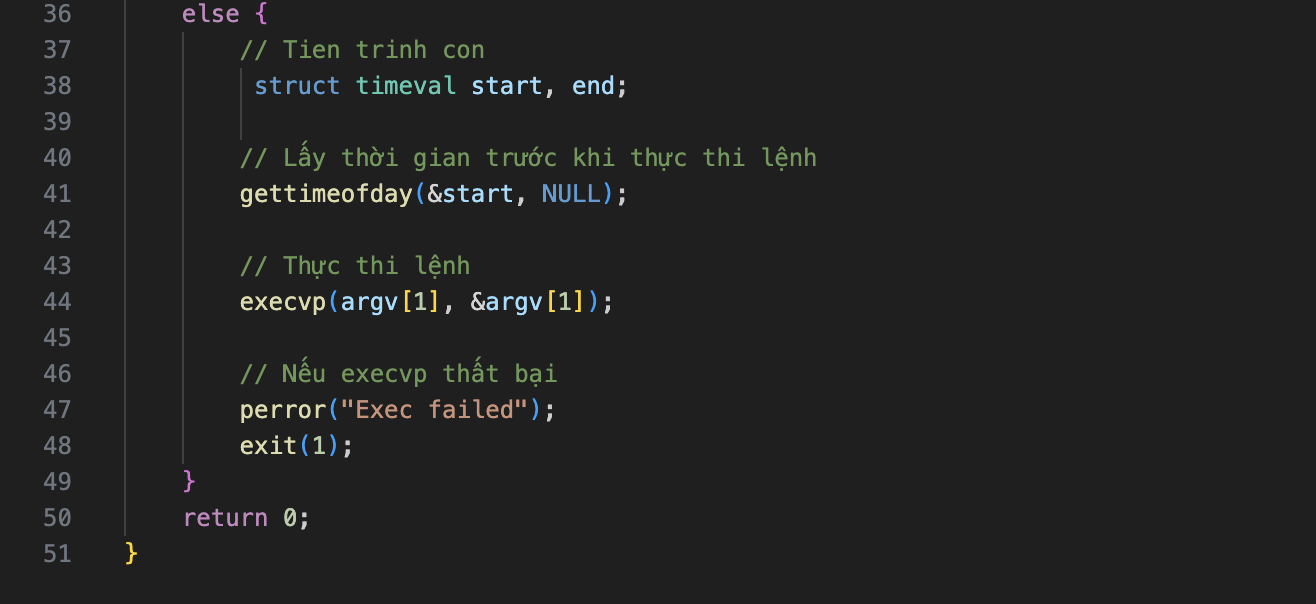
# 2. Viết chương trình time.c thực hiện đo thời gian thực thi

# 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.

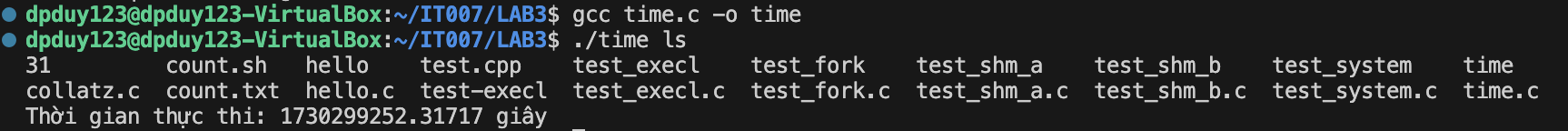
Code:



Giải thích code:

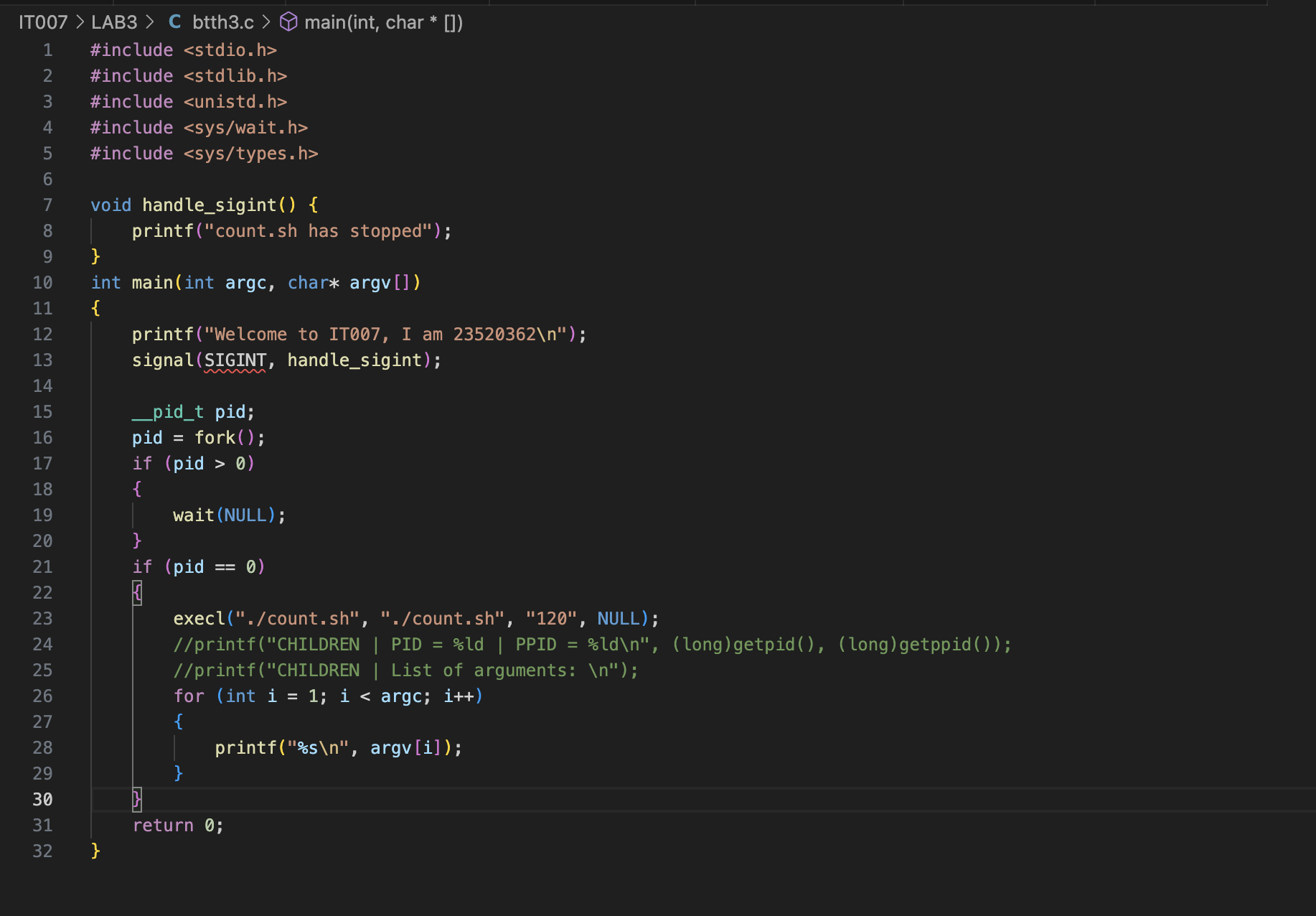
1. **Thư viện bao gồm**:
   * stdio.h: Để sử dụng các hàm nhập/xuất.
   * stdlib.h: Để sử dụng các hàm chung như exit().
   * unistd.h: Để sử dụng fork() và exec().
   * sys/types.h và sys/wait.h: Để làm việc với tiến trình.
   * sys/time.h: Để sử dụng gettimeofday().
2. **Kiểm tra tham số đầu vào**:
   * Nếu số lượng tham số nhỏ hơn 2, in ra thông báo hướng dẫn và kết thúc chương trình.
3. **Tạo tiến trình con**:
   * Sử dụng fork() để tạo một tiến trình con.
4. **Trong tiến trình con**:
   * Sử dụng gettimeofday() để lấy thời gian bắt đầu.
   * Gọi execvp() để thực thi lệnh shell. execvp() cho phép bạn truyền danh sách các tham số.
   * Nếu execvp() thất bại, in ra thông báo lỗi và kết thúc tiến trình con.
5. **Trong tiến trình cha**:
   * Sử dụng wait() để chờ tiến trình con hoàn thành.
   * Sau khi tiến trình con kết thúc, lấy thời gian kết thúc bằng gettimeofday().
   * Tính toán thời gian thực thi bằng cách lấy hiệu giữa thời gian kết thúc và thời gian bắt đầu.
   * In ra thời gian thực thi.

Biên dịch và thực thi:

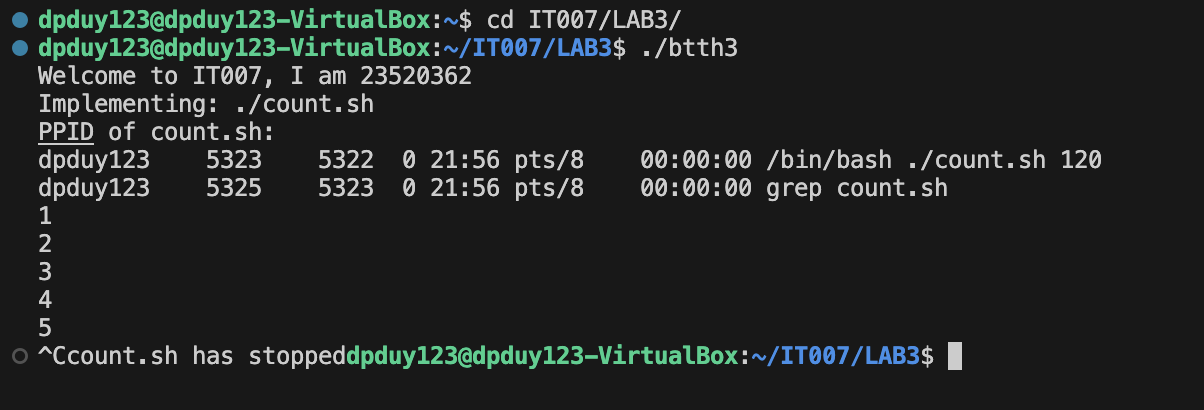


# 3.

Code:



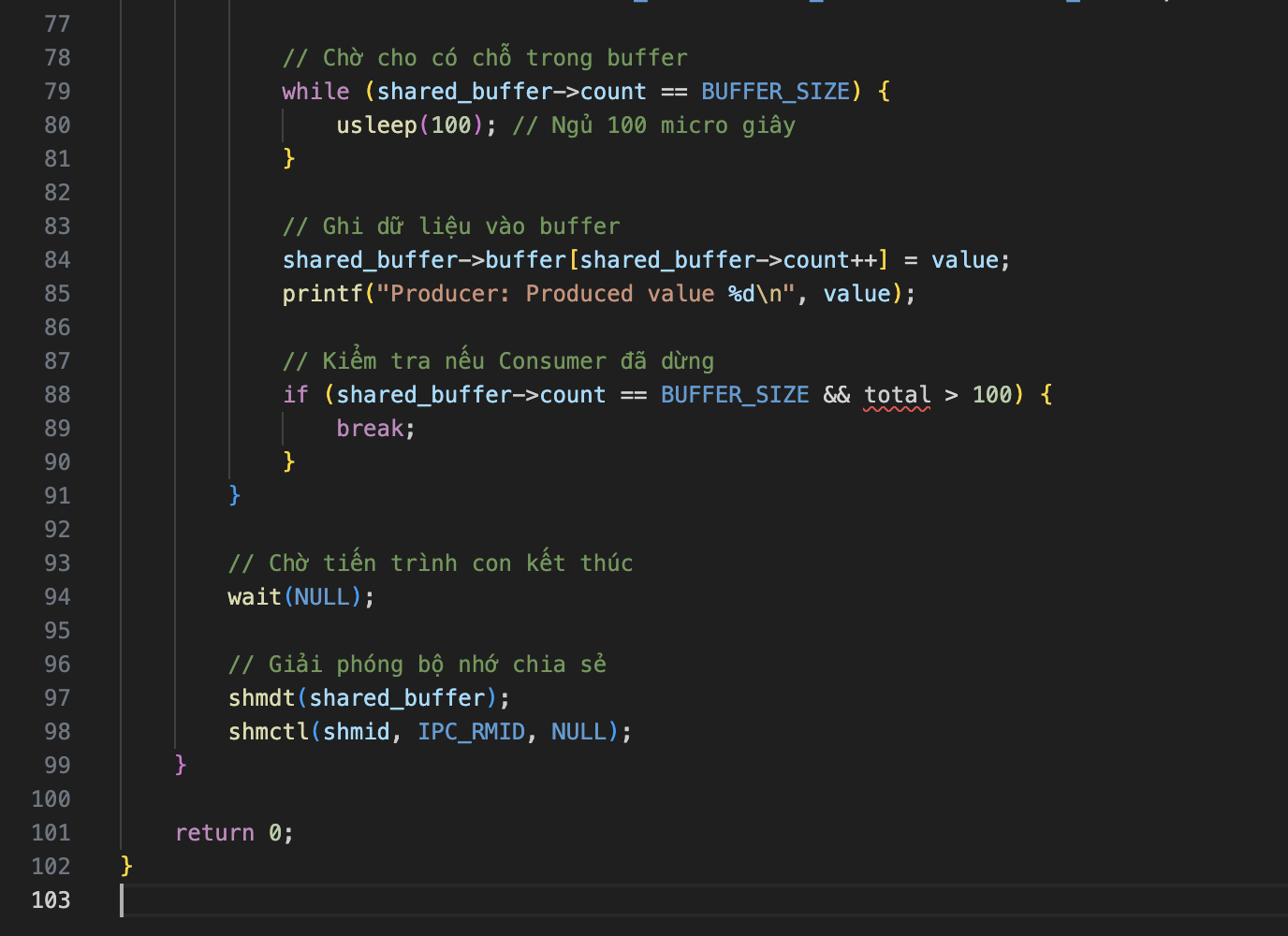
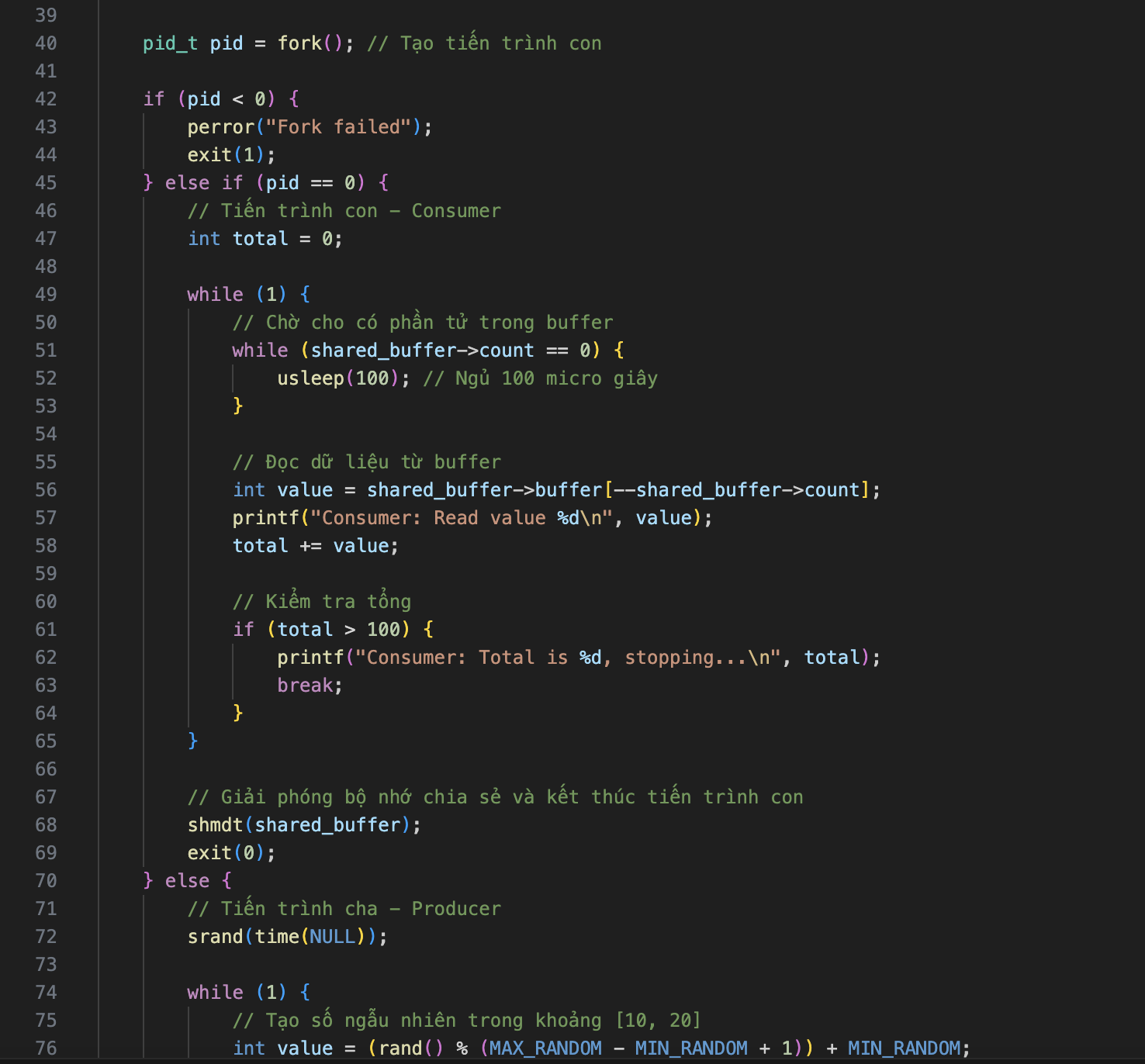
Biên dịch:



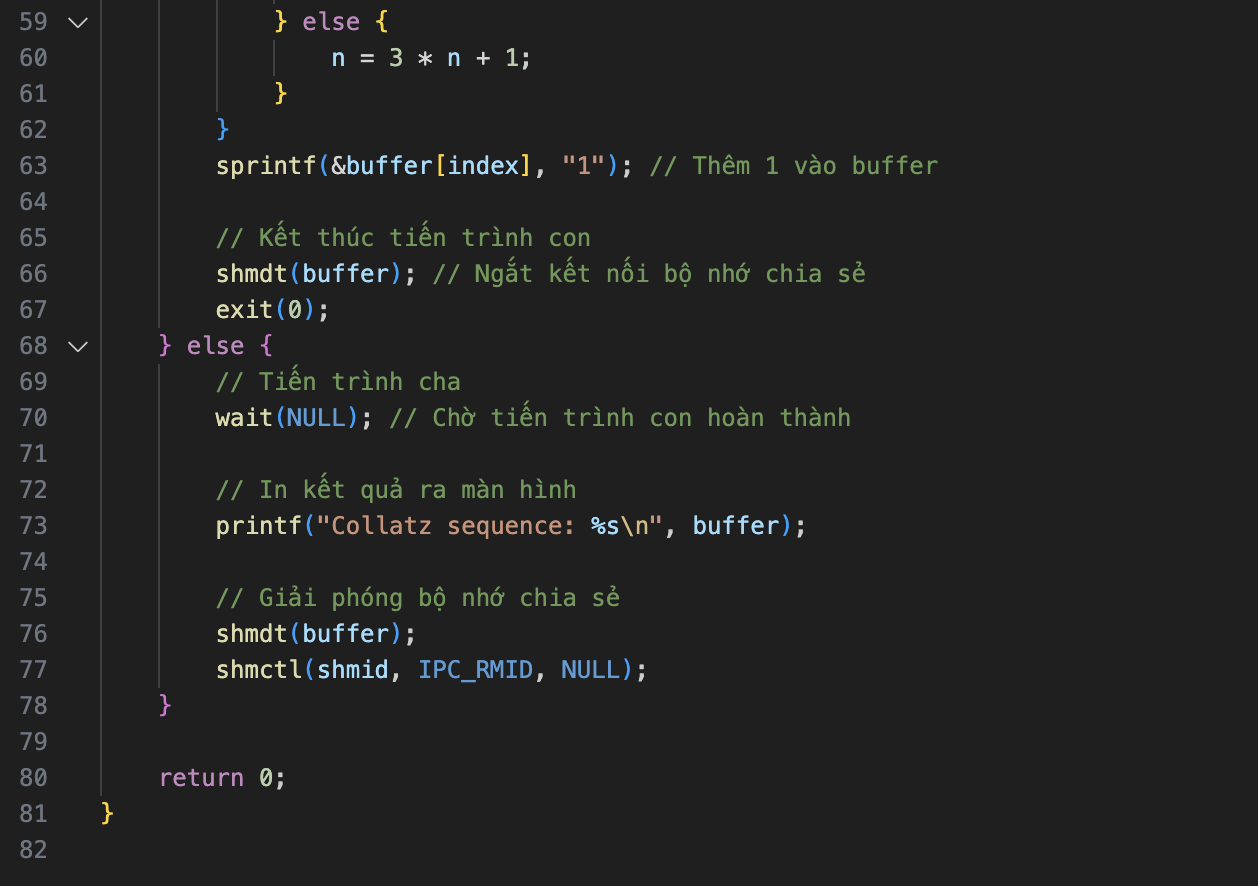
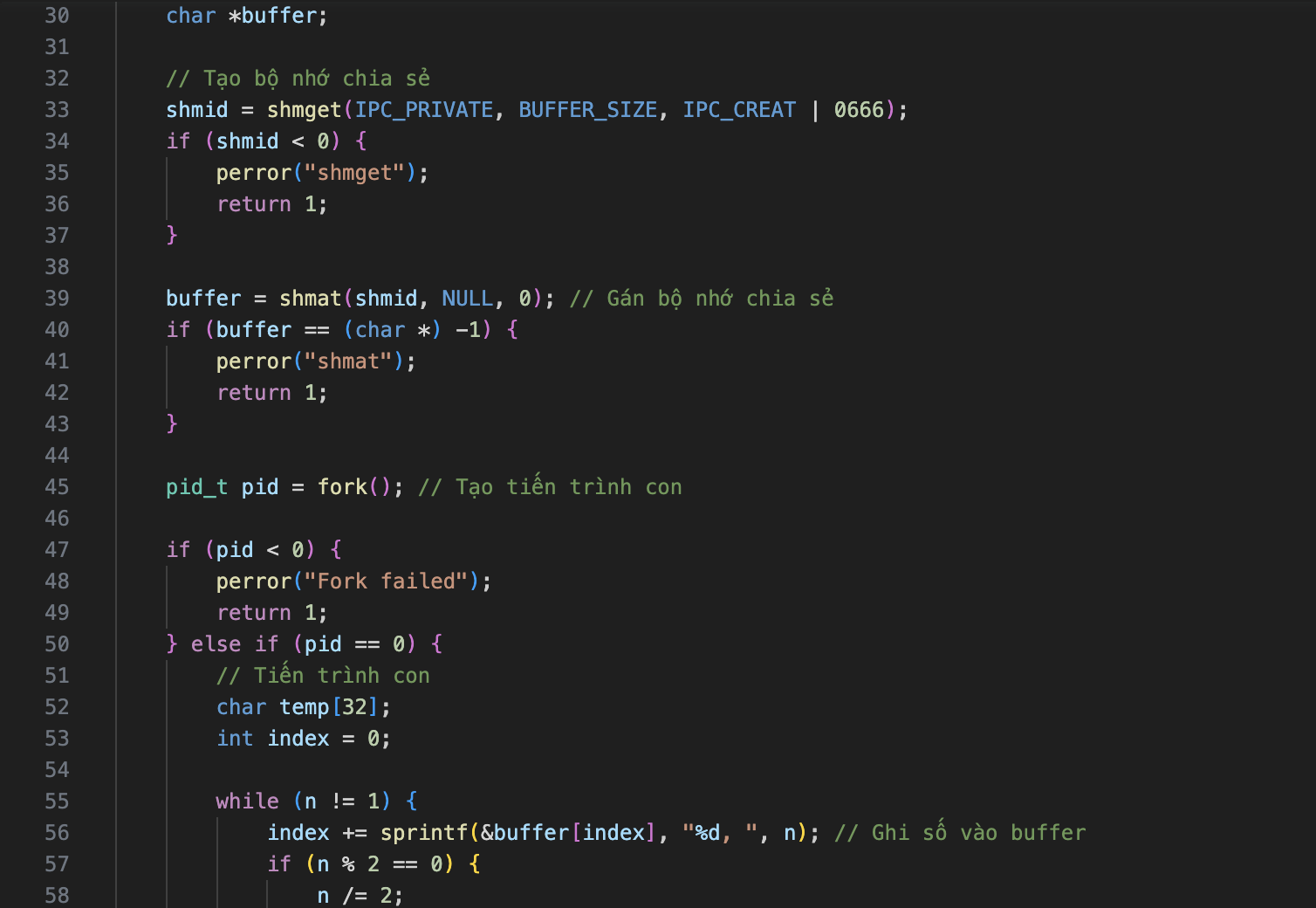
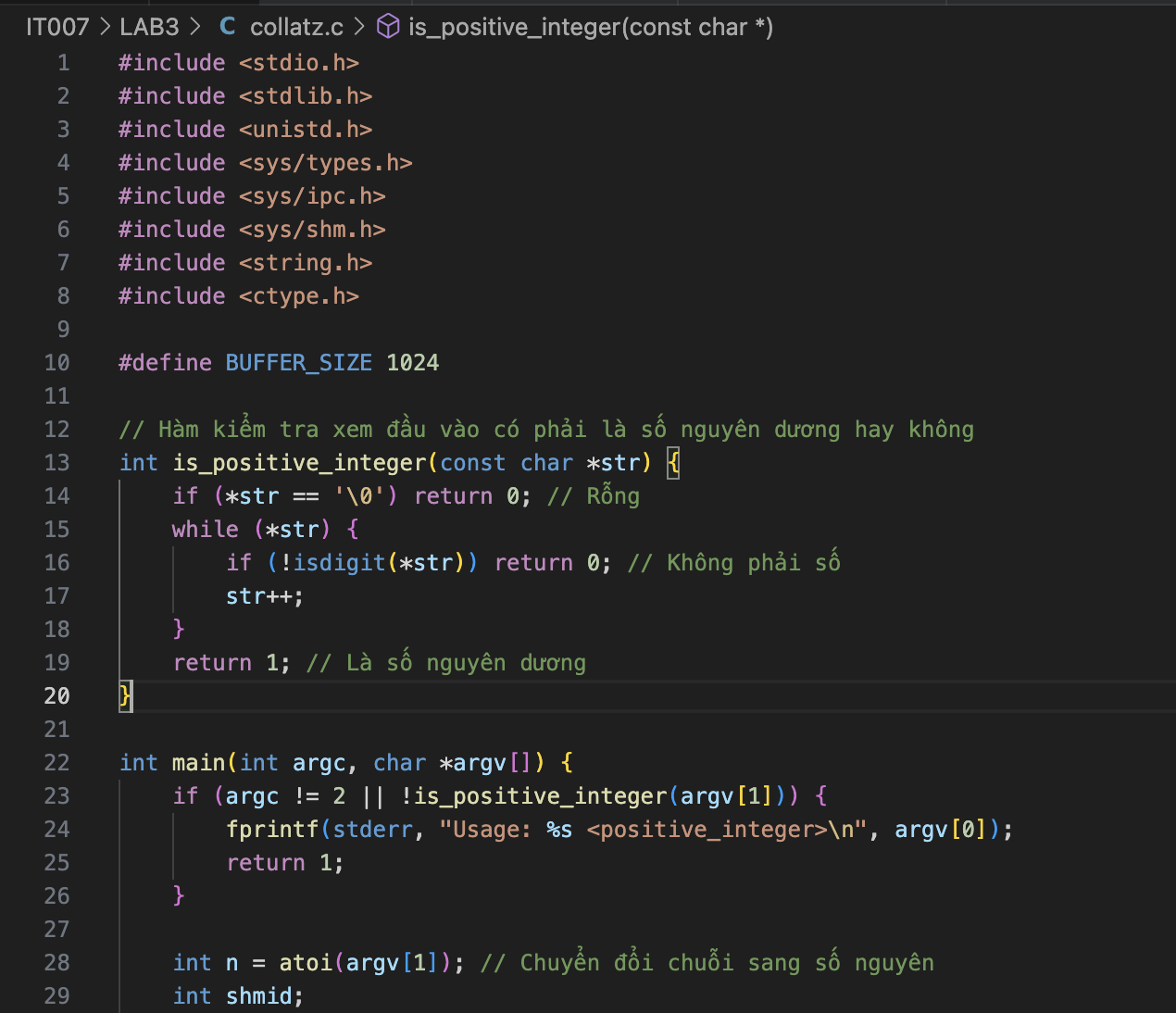
Ở đây vì thời gian có hạn nên em Ctrl C khi count đếm đến 5

# 4. Viết chương trình mô phỏng bài toán Producer – Consumer:

Code:



# Bài tập ôn tập:



Giải thích code:

 **Hàm kiểm tra số nguyên dương**:

* Hàm is\_positive\_integer() kiểm tra xem chuỗi đầu vào có phải là số nguyên dương không. Nếu không, chương trình sẽ in thông báo lỗi và thoát.

 **Bộ nhớ chia sẻ**:

* Tạo bộ nhớ chia sẻ bằng shmget() và gán nó vào buffer sử dụng shmat().

 **Tiến trình con**:

* Tiến trình con tính toán chuỗi Collatz từ số n. Mỗi giá trị được ghi vào buffer sử dụng sprintf().
* Khi n trở về 1, tiến trình con kết thúc và ngắt kết nối bộ nhớ chia sẻ.

 **Tiến trình cha**:

* Tiến trình cha chờ cho tiến trình con hoàn thành bằng wait(), sau đó in ra chuỗi Collatz từ buffer.
* Giải phóng bộ nhớ chia sẻ bằng shmdt() và shmctl().

Biên dịch và kiểm thử các testcase:

