|  |
| --- |
| Name: Nguyễn Huỳnh Nhân  ID: 23521080  Class:IT007.P28.2 |

OPERATING SYSTEM  
LAB X’S REPORT

**SUMMARY**

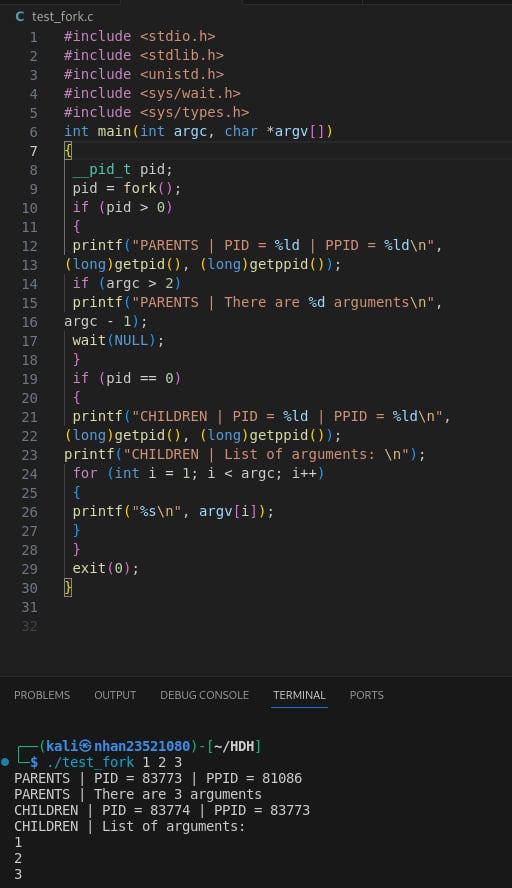
|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | | **Status** | **Page** |
| Section 3.5 | Ex 1 | Hoàn thành | 2 |
| Ex 2 | Hoàn thành | 8 |
| Ex 3 | Hoàn thành | 10 |
| EX 4 | Hoàn thành | 11 |
|  |  |  |
|  |  |  |
|  |  |  |
| … | … |  |  |
| … |  |  |

**Self-scrores: 10/10**

# Section 3.5

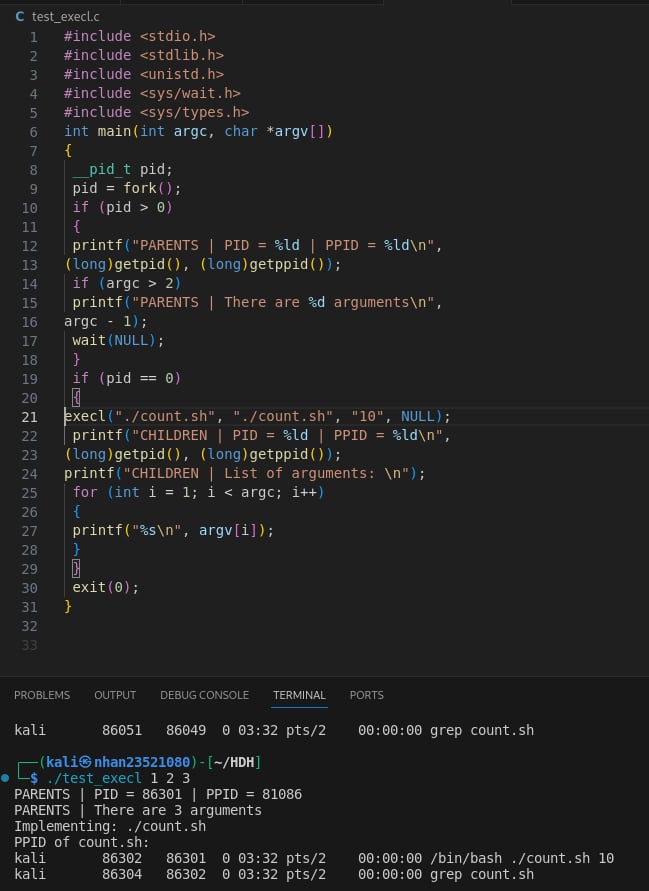
## Thực hiện Ví dụ 3-1, Ví dụ 3-2, Ví dụ 3-3, Ví dụ 3-4 giải thích code và kết quả nhận được?

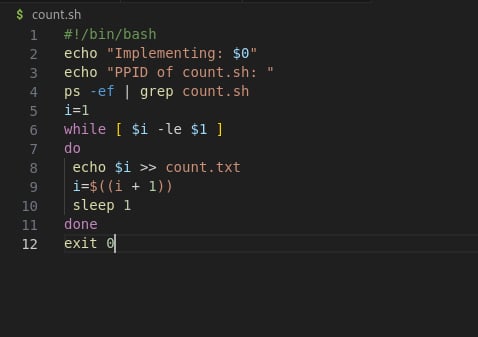
VÍ DỤ 3-1

 + Giải thích  
  
+ Giải thích:

* Hàm fork() tạo ra bản sao của tiến trình gọi nó.Trong trường hợp trên nó tạo ra một tiến trình con y hệt cha
* Nếu pid > 0 bắt đầu xử lý tiến trình cha,chương trình sẽ in ra PID và PPID của tiến trình cha. Nếu số lượng tham số nhập vào lớn hơn 2 thì sẽ in ra số lượng tham số.Sau đó tiến trình cha chờ tiến trình con kết thúc bằng lệnh wait(NULL);
* Nếu pid=0 bắt đầu xử lý tiến trình con, chương trình sẽ in ra pid và ppid của tiến trifnh con,sau đó in ra các tham số đã truyền vào

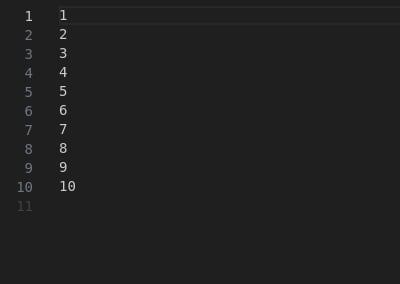
Ví dụ 3-2



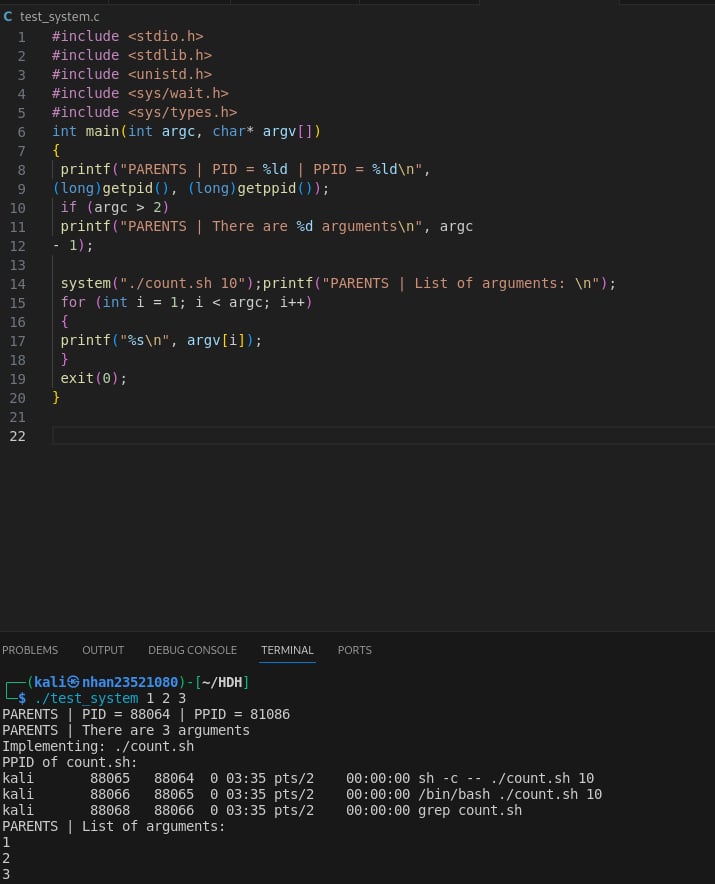


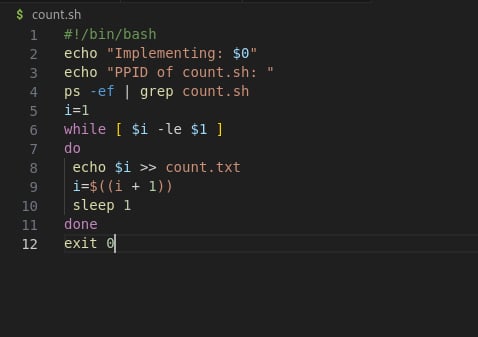
Giải thích:

* Sử dụng hàm fork() để tạo tiến trình con.
* 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" và ghi đè lên tiến trình con đang chạy bên dưới . In ra thông tin về PID và PPID của tiến trình con. In ra các tham số dòng lệnh truyền vào.
* Hàm coun.sh dùng để in ra từ 1->$1 sau mỗi 1s



Ví dụ 3-3

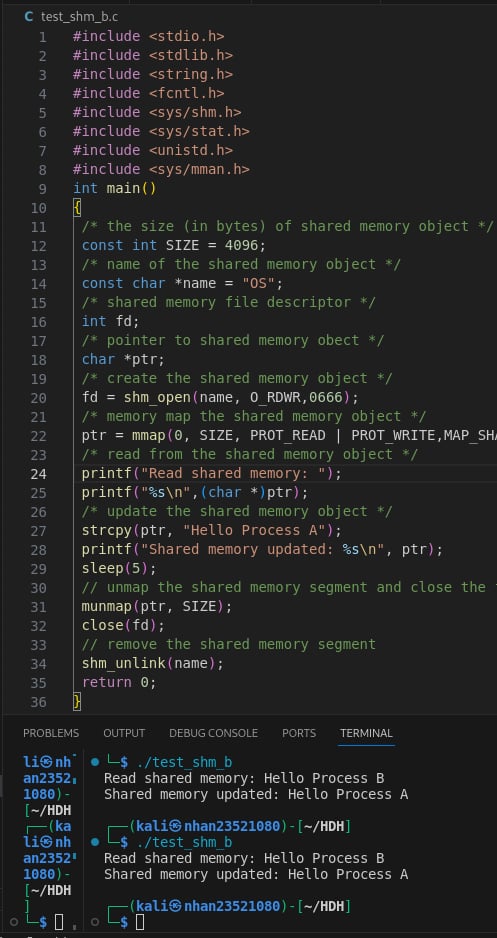


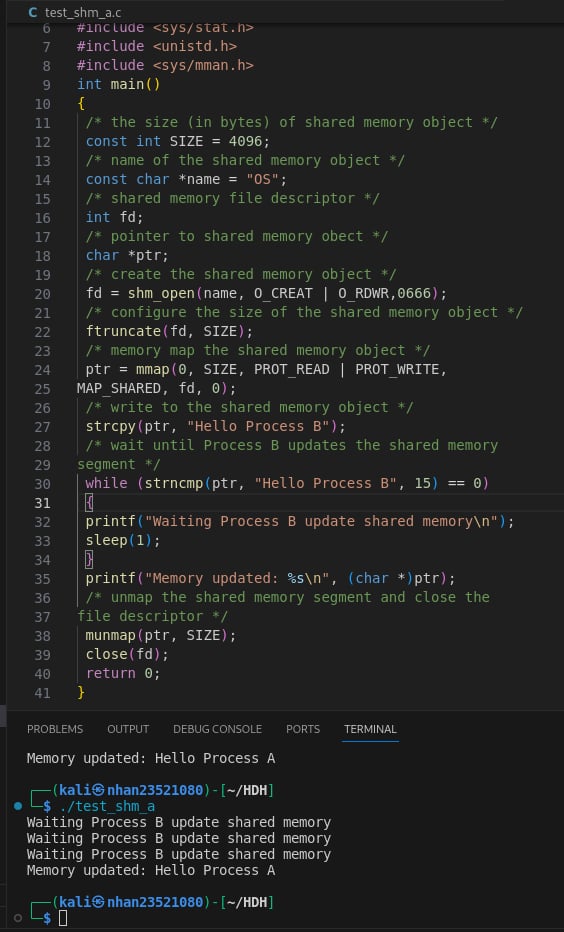


Giải thích

* 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 và sẽ tạo mới hoàn toàn tiến trình

Ví dụ 3-4





Giải thích:

* Process A: tạo một đối tượng shared memory, ghi dữ liệu vào và sau đó chờ tiến trình B cập nhật dữ liệu vào đối tượng shared memory,khi B cập nhật xong A sẽ đọc dữ liệu và in dữ liệu ra
* Tạo một đối tượng mới bàng shm\_open().
* Cấu hình kích thước bằng ftruncate().
* Ánh xạ đối tượng shared memory vào không gian địa chỉ của quy trình bằng mmap().
* Ghi dữ liệu vào share memory bằng cách sao chép chuỗi “Hello Process B” vào con trỏ ptr
* Chờ đợi B cập nhật dữ liệu trong 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**

* Process B mở đối tượng shared memory được tạo bởi A đọc dữ liệu và thay đổi đối tượng shared memory sau đó chờ một thời gian để kết thúc
* 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 filebằng munmap() và close().
* Xóa đối tượng shared memory bằng hàm shm\_unlink().

## 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:

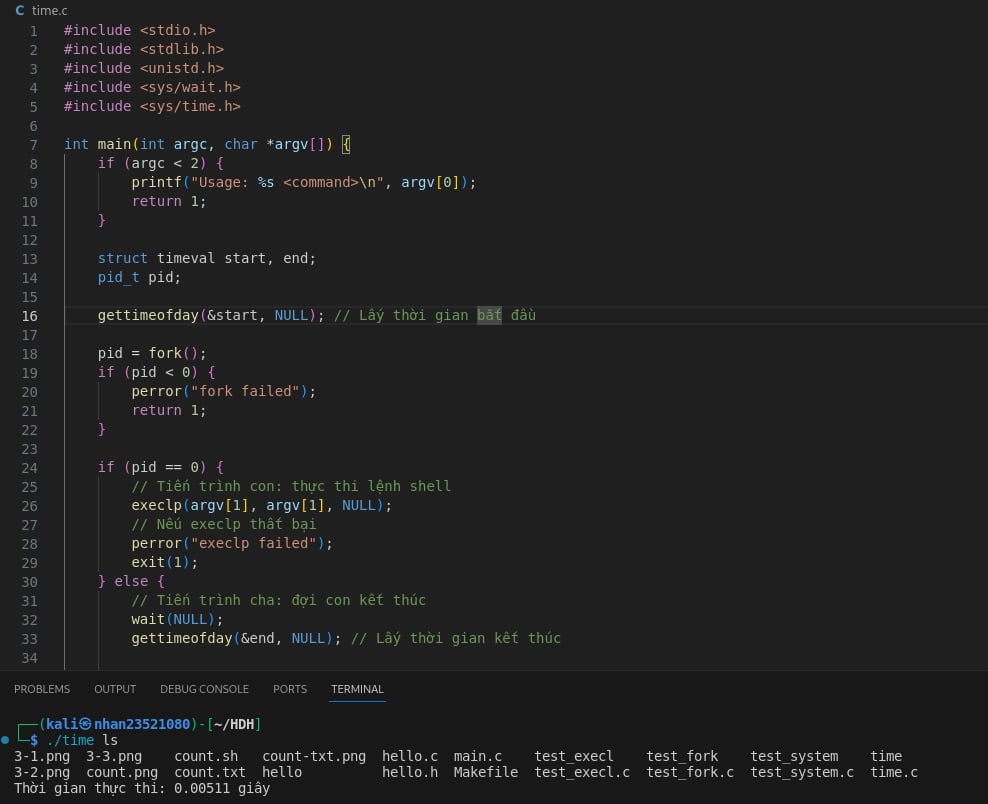
$ ./time ls

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.**



double elapsed = (end.tv\_sec - start.tv\_sec) +

(end.tv\_usec - start.tv\_usec) / 1000000.0;

printf("Thời gian thực thi: %.5f giây\n", elapsed);

Giải thích

* Nếu tham số nhận vào < 2 thì sẽ cảnh báo và thoát
* Gọi struct start và end có kiểu dữ liệu timeval để lấy thời gian bắt đầu và kết thúc,getimeofday(&start,null) để lấy thời gian ban đầu
* Tạo tiến trình con bằng fork(),fork<0 tạo tiến trình con thất bại và báo lỗi , fork()=0 tiến trình con đang chạy, sử dụng execlp để lấy tham số truyền vào
* Fork>0 tiến trình cha đang chạy tiến trình cha chờ tiến trình con chạy xong rồi mới gọi hàm gettimeofday(&end,null) để tính thời gian kết thúc
* Thời gian chạy bằng thời gian kết thúc - thời gian ban đầu

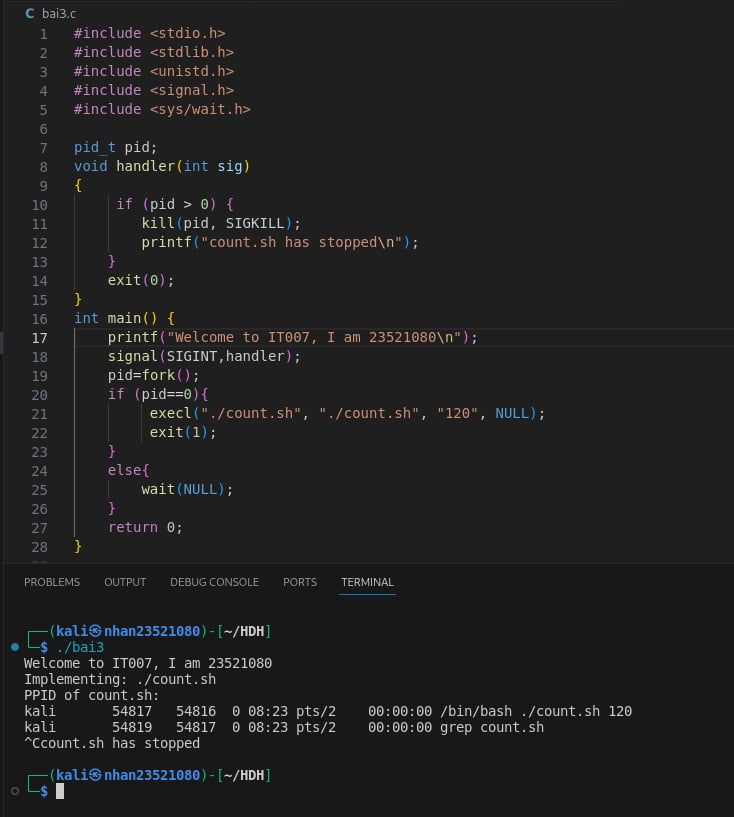
# 3.Viết một chương trình làm bốn công việc sau theo thứ tự:

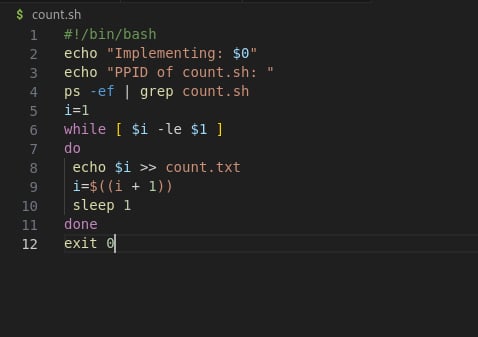
# In ra dòng chữ: “Welcome to IT007, I am <your\_Student\_ID>!”

# 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





Giải thích

* Tạo hàm handler để bắt sự kiện khi nhấn ctrl+C: kết thúc tiến trình và thông báo ra màn hình
* Signal(SIGINT,handler): hàm gọi sự kiện khi nhấn ctrl+C thì sẽ gọi đến hàm handler
* In ra dòng chữ chào mừng với Student ID.
* 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.

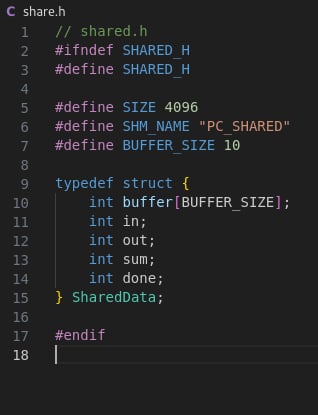
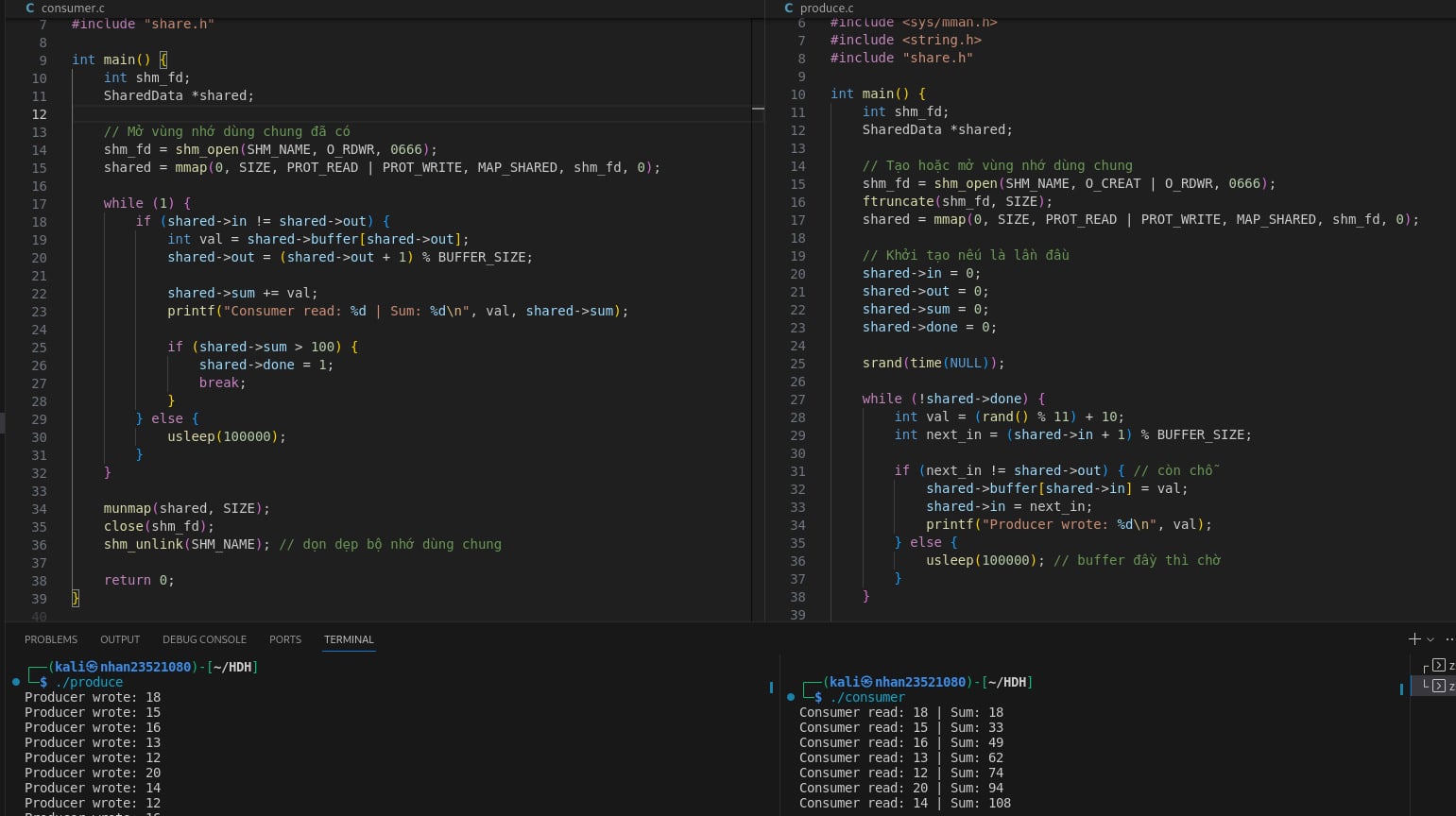
# 4.Viết chương trình mô phỏng bài toán Producer - Consumer như sau:

# 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



Giải thích:

Process Produce.c

* Share.h dùng để tạo các biến dùng chung cho cả 2 tiến trình producer và consumer
* Tiến trình produce khởi tạo vùng nhớ dùng chung PC\_SHARED
* Share->in : vị trí hiện tại của producer
* Share-out vị trí hiện tại của consumer
* Share-done: biến cho biết tiến trình con đã hoàn thành
* Share->sum: biến tính tổng
* Srand(time(NULL)): dùng để tạo các số ngẫu nhiên không trùng lặp
* Kiểm tra nếu tiến trình con chưa hoàn thành thì thực hiện while
* Gán biến val có giá trị từ 10-20
* Next->in: vị trí tiếp theo cuả producer
* Kiểm tra nếu vị trí tiếp theo và vị trí hiện tại của consumer khác nhau (còn chỗ) thì thêm giá trị val vào buffer hiện tại sau đó tăng vị trí hiện tại của producer lên 1 ngược lại thì chờ

Process Consumer.c

* Khởi tạo biến dùng chung PC\_SHARED
* Cộng dồn giá trị của sum cho đến khi >100 thì gán share->done=1(kết thúc tiến trình )
* Giải phóng và hủy bỏ tiến trình
* Dọn dẹp bộ nhớ dùng chung

…