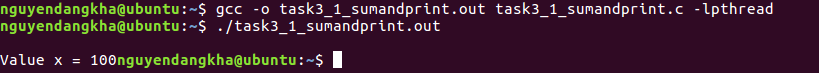
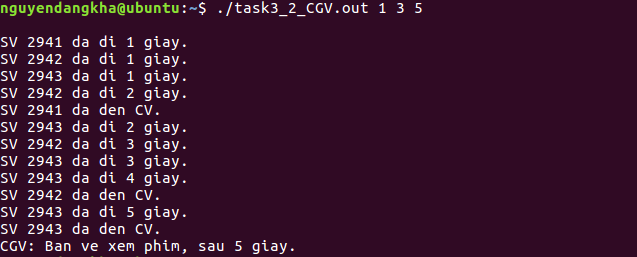
**Lab 7.3**

*Yêu cầu 1:*



*Yêu cầu 2:*



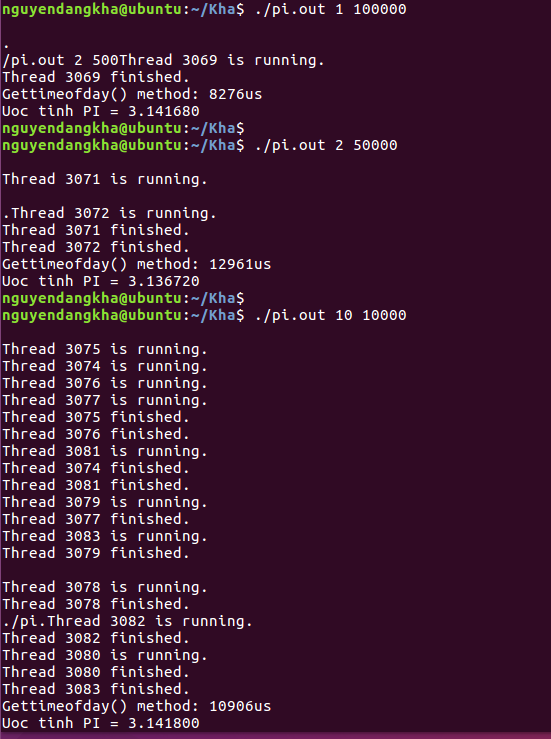
*Bài tập 2:*

Hàng rào bộ nhớ (pthread\_barrier) **không phù hợp** để đồng bộ hóa trong bài toán Producer-Consumer vì:

* Nó không hỗ trợ đồng bộ hóa dựa trên điều kiện (bộ đệm rỗng/đầy).
* Nó yêu cầu tất cả các tiểu trình chờ nhau tại một điểm cố định, không phù hợp với tính chất không đồng bộ của producer và consumer.
* Nó không bảo vệ được bộ đệm khỏi truy cập đồng thời.

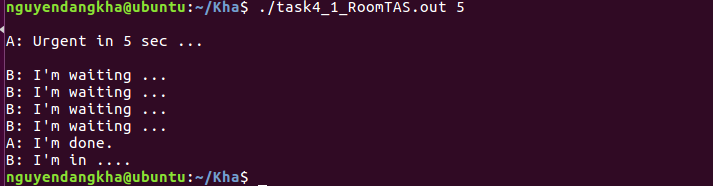
Thay vào đó, nên sử dụng **mutex** kết hợp với **condition variables** hoặc **semaphores** để đảm bảo đồng bộ hóa chính xác và hiệu quả trong bài toán Producer-Consumer.

*Bài tập 1:*

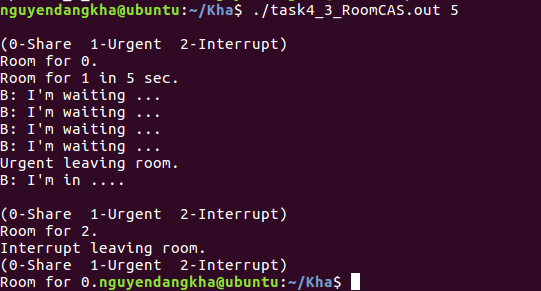


**Lab 7.4:**

*Yêu cầu 1:*

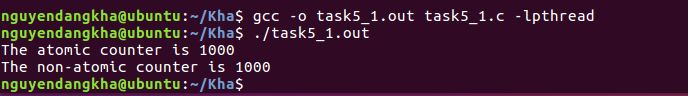


*Yêu cầu 3:*

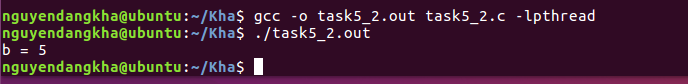


**Lab 7.5:**

*Yêu cầu 1:*



*Yêu cầu 2:*



**Biến số đơn nguyên** (atomic variable) là một loại biến mà các thao tác đọc, ghi, hoặc cập nhật (như tăng, giảm, so sánh và trao đổi) được thực hiện một cách **nguyên tử**. Nghĩa là thao tác này được hoàn thành trong **một bước duy nhất**, không thể bị gián đoạn hoặc xen kẽ bởi các luồng/tiến trình khác.

Trong hệ điều hành, biến số đơn nguyên được sử dụng để:

* Tránh **race condition** (tình trạng đua dữ liệu) khi nhiều luồng truy cập cùng một biến.
* Đảm bảo tính **nhất quán** (consistency) của dữ liệu mà không cần sử dụng các cơ chế khóa phức tạp (như semaphore hoặc mutex) trong một số trường hợp.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Biến thông thường** | **Biến số đơn nguyên** |
| **Tính nguyên tử** | Không đảm bảo tính nguyên tử. Thao tác (như counter++) có thể bị chia thành nhiều bước (đọc, tăng, ghi), dễ bị gián đoạn bởi luồng khác. | Đảm bảo tính nguyên tử. Thao tác được thực hiện trong một bước duy nhất, không thể bị gián đoạn. |
| **Race condition** | Dễ xảy ra race condition khi nhiều luồng truy cập cùng lúc mà không có cơ chế bảo vệ (như mutex). | Loại bỏ race condition cho các thao tác nguyên tử, không cần khóa trong một số trường hợp. |
| **Hiệu suất** | Thường cần sử dụng khóa (mutex, semaphore) để bảo vệ, dẫn đến overhead (chi phí) lớn hơn. | Hiệu suất cao hơn vì sử dụng các lệnh cấp thấp của CPU (như test-and-set, compare-and-swap). |
| **Phạm vi sử dụng** | Phù hợp cho các trường hợp không có tranh chấp hoặc được bảo vệ bởi cơ chế đồng bộ. | Dành cho các thao tác đơn giản (tăng, giảm, đọc, ghi) trong môi trường đa luồng. |
| **Cơ chế triển khai** | Không có cơ chế đặc biệt, chỉ là vùng nhớ thông thường | Được triển khai bằng các lệnh nguyên tử của phần cứng hoặc thư viện (như <atomic> trong C++). |
| **Phức tạp khi sử dụng** | Có thể phức tạp hơn vì cần thêm cơ chế đồng bộ (mutex, semaphore). | Đơn giản hơn cho các thao tác nguyên tử, nhưng giới hạn ở các thao tác được hỗ trợ. |

Lab 8.5:

Yêu cầu 1:

