**BÀI THỰC HÀNH BUỔI 4**

**BT1. Tạo và Chạy Threads Cơ Bản**

Viết một chương trình sử dụng thư viện pthread.h để thực hiện các thao tác sau:

1. Tạo hai threads. Mỗi thread sẽ in ra một chuỗi thông báo kèm theo ID của thread (ví dụ: "Thread 1: Hello from thread").
2. Sử dụng hàm pthread\_join để đợi cả hai threads hoàn thành trước khi chương trình kết thúc.
3. Sử dụng pthread\_create và pthread\_join như thế nào? Khi nào thread kết thúc?

Gợi ý: Sử dụng hàm pthread\_create để tạo threads và pthread\_join để chờ các threads hoàn thành.

**BT2. Đồng Bộ Hóa Threads với Mutex**

Viết một chương trình sử dụng pthread và mutex để thực hiện các thao tác sau:

1. Tạo một biến toàn cục counter khởi tạo bằng 0.
2. Tạo ba threads, mỗi thread tăng giá trị của counter lên 1, 1.000.000 lần.
3. Sử dụng mutex để đảm bảo rằng việc tăng counter được thực hiện an toàn.
4. In ra giá trị cuối cùng của counter khi tất cả các threads hoàn thành.
5. Hỏi: Tại sao cần mutex trong bài này? Điều gì xảy ra nếu bỏ mutex?

Gợi ý: Sử dụng pthread\_mutex\_lock và pthread\_mutex\_unlock để khóa và mở khóa mutex khi truy cập vào counter.

**BT3. Sử Dụng Condition Variables**

Viết một chương trình có hai threads: producer và consumer như sau:

1. Producer: Tạo ra một số ngẫu nhiên từ 1 đến 10 và lưu vào một biến toàn cục data.
2. Consumer: Đợi producer cung cấp dữ liệu, sau đó đọc và in ra dữ liệu đó.
3. Sử dụng pthread\_cond\_wait và pthread\_cond\_signal để đồng bộ hóa giữa producer và consumer, đảm bảo rằng consumer chỉ đọc dữ liệu khi producer đã cung cấp xong.
4. Lặp lại quá trình trên 10 lần và in ra tất cả các giá trị do consumer đọc được.

Gợi ý: Sử dụng pthread\_cond\_wait để cho consumer đợi cho đến khi có tín hiệu từ producer rằng dữ liệu đã sẵn sàng.

**BT4. Đếm Số Chẵn và Số Lẻ với Threads**

Viết một chương trình để thực hiện các bước sau:

1. Tạo một mảng ngẫu nhiên chứa 100 số nguyên từ 1 đến 100.
2. Tạo hai threads: một thread đếm số chẵn và một thread đếm số lẻ trong mảng.
3. Khi hai threads hoàn tất việc đếm, in ra tổng số số chẵn và tổng số số lẻ.
4. Sử dụng pthread\_join để đảm bảo rằng chương trình chỉ kết thúc khi cả hai threads hoàn thành nhiệm vụ.

Gợi ý: Mỗi thread sẽ xử lý một phần công việc khác nhau, vì vậy không cần mutex trong bài này.

**BT5. Chia Sẻ Tài Nguyên Với Read-Write Lock**

Viết một chương trình mô phỏng tình huống có nhiều thread đọc và ghi dữ liệu:

1. Tạo một biến data chứa một số nguyên.
2. Tạo 5 threads đọc và 2 threads ghi:
   * Các thread đọc sẽ chỉ đọc data mà không thay đổi.
   * Các thread ghi sẽ tăng giá trị của data lên 1.
3. Sử dụng read-write lock để đảm bảo nhiều threads có thể đọc cùng lúc, nhưng chỉ một thread được ghi tại bất kỳ thời điểm nào.
4. In ra giá trị của data sau khi tất cả các threads hoàn thành.

Gợi ý: Sử dụng pthread\_rwlock\_rdlock và pthread\_rwlock\_wrlock cho đọc-ghi đồng bộ.

**BT6. Tính Tổng Mảng với Threads và Mutex**

Viết một chương trình tính tổng một mảng lớn gồm 1 triệu số nguyên.

1. Chia mảng thành 4 phần bằng nhau.
2. Tạo 4 threads, mỗi thread tính tổng một phần của mảng.
3. Sử dụng một biến tổng toàn cục và mutex để tổng hợp kết quả từ tất cả các threads.
4. In ra kết quả tổng của mảng sau khi các threads hoàn thành.

Gợi ý: Sử dụng pthread\_mutex\_lock để bảo vệ biến tổng khi các threads cộng kết quả của chúng.