|  |
| --- |
| Họ và tên: Phan Anh Khôi  Mã số sinh viên: 23520787  Lớp:IT007.P23.1 |

HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 5

**CHECKLIST**

**5.5. BÀI TẬP THỰC HÀNH**

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

**5.6. BÀI TẬP ÔN TẬP**

|  |  |
| --- | --- |
|  | **BT 1** |
| **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:* ***<Tên nhóm>\_LAB5.pdf***

**5.5. BÀI TẬP THỰC HÀNH**

# Hiện thực hóa mô hình trong ví dụ 5.3.1.2, tuy nhiên thay bằng điều kiện sau: sells <= products <= sells + [4 số cuối của MSSV]

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Sử dụng 2 thread cho 2 process A, B

Sử dụng 2 semaphore cho 2 ràng buộc đề bài yêu cầu

Sử dụng 1 mutex để bảo vệ các biến chia sẻ (products,sell)

Thread A là bên bán, Thread B là bên sản xuất

Cách làm:

Ngoài hàm main: khởi tạo biến product ( số lượng sản phẩm tồn kho), sell (số lượng sản phẩm đã bán được), khởi tạo 2 biến semaphore và một biến mutex

Trong hàm main:

Tạo 2 thread A và B Gán giá trị cho mutex

Gán giá trị bằng 0 đối với semaphore cho ràng buộc sell<=products (sem1)

Gán giá trị bằng 787 ( MSSV ) đối với semaphore cho ràng buộc products<=sell+MSSV (sem2) Hợp 2 tiểu trình để hàm main chờ cho 2 tiểu trình thực thi

Thread A:

1.Chạy vòng lặp

2.Sem1 wait kiểm tra xem có sản phẩm để bán không, nếu không thì chờ Thread B tạo ra sản phẩm

3.Mutex lock để bảo vệ các biến chia sẻ

4. Tăng biến sell và giảm biến product 1 đơn vị

5. Xuất ra số sản phẩm đã bán

6. Mutex unlock

7. Sem2 post để tăng biến sem2 1 đơn vị cũng như sản phẩm đã bán được cộng 1 Thread B:

1. Tạo biến ProductID để đánh số thứ tự cho product

2. Chạy vòng lặp

3. Sem2 wait kiểm tra xem số sản phẩm bán được có <= số sản phẩm đã bán + 1046 hay không, nếu không thì chờ Thread A bán sản phẩm

4. Mutex lock để bảo vệ các biến chia sẻ

5. Tăng biến product 1 đơn vị

6. Xuất ra số lượng sản phẩm tồn kho và thông báo đã sản xuất ra sản phẩm mới

7. Mutex unlock

8. Sem1 post để thông báo cho Thread A có sản phẩm để bán

A screen shot of a computer

AI-generated content may be incorrect.

Thread B tạo ra sản phầm in ra số thứ tự của sản phẩm và số lượng sản phầm có thể bán được ( Các dòng dài) Thread A bán sản phẩm in ra số lượng sản phẩm đã bán được cũng như là số thứ tự của sản phẩm vừa bán ( Các dòng ngắn)

# Cho một mảng a được khai báo như một mảng số nguyên có thể chứa n phần tử, a được khai báo như một biến toàn cục. Viết chương trình bao gồm 2 thread chạy song song:

* Một thread làm nhiệm vụ sinh ra một số nguyên ngẫu nhiên sau đó bỏ vào a. Sau đó đếm và xuất ra số phần tử của a có được ngay sau khi thêm vào.
* Thread còn lại lấy ra một phần tử trong a (phần tử bất kỳ, phụ thuộc vào người lập trình). Sau đó đếm và xuất ra số phần tử của a có được ngay sau khi lấy ra, nếu không có phần tử nào trong a thì xuất ra màn hình “Nothing in array a”.

Chạy thử và tìm ra lỗi khi chạy chương trình trên khi chưa được đồng bộ. Thực hiện đồng bộ hóa với semaphore.

A computer screen shot of a program code

AI-generated content may be incorrect.

A computer code with white text

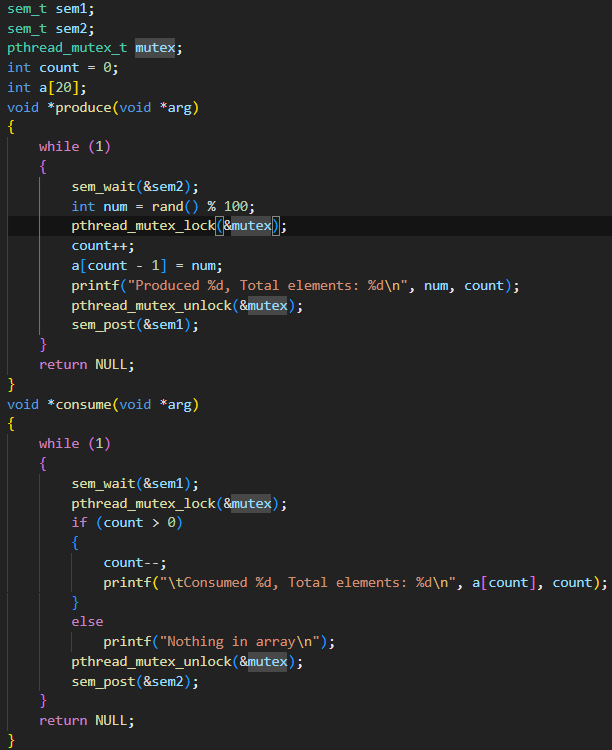
AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Khi không đồng bộ hóa:

Kết quả khi không đồng bộ hóa: thấy mảng đang chứa 154 phần tử. Nhưng ở dòng tiếp theo output bên consumer lại thông báo rằng trong mảng không có phần tử nào. Điều này là do consumer truy cập vào biến count trong khi producer đang thay đổi dữ liệu trên biến count dẫn đến lỗi.

Đồng bộ hóa bằng semaphore và mutex

Lúc này không còn tình trạng như trên do các biến duẽ liệu đã được bảo vệ không cho cả 2 bên cùng truy cập:

A screenshot of a computer

AI-generated content may be incorrect.

# Cho 2 process A và B chạy song song như sau:

|  |  |
| --- | --- |
| int x = 0; | |
| **PROCESS A** | **PROCESS B** |
| processA()  {  while(1){  x = x + 1;  if (x == 20)  x = 0;  print(x);  }  } | processB()  {  while(1){  x = x + 1;  if (x == 20)  x = 0;  print(x);  }  } |

Hiện thực mô hình trên C trong hệ điều hành Linux và nhận xét kết quả.

A screen shot of a computer program

AI-generated content may be incorrect.

Kết quả

A screenshot of a computer

AI-generated content may be incorrect.

Khi không đồng bộ hóa 2 tiến trình output có lúc sẽ sai.

Ở dưới dòng A=11 ta mong đợi kết quả là A:12. Nhưng output lại là B:7, điều này có thể hiểu là Process A truy cập biến x lúc x đang có giá trị là 11 rồi tăng giá trị lên 12. Nhưng chưa xuất ra màn hình cho đến khi biến x được reset và tăng giá trị lên 1 bởi B.

# Đồng bộ với mutex để sửa lỗi bất hợp lý trong kết quả của mô hình Bài 3.

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Kết quả:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Kết quả bây giờ khi sử dụng mutex không còn tình trạng như câu trước, biến x bâu giờ chỉ cho phép 1 thread truy cập**

**5.6. BÀI TẬP ÔN TẬP**

# Biến ans được tính từ các biến x1, x2, x3, x4, x5, x6 như sau:

w = x1 \* x2; (a)

v = x3 \* x4; (b)

y = v \* x5; (c)

z = v \* x6; (d)

y = w \* y; (e)

z = w \* z; (f)

ans = y + z; (g)

Giả sử các lệnh từ (a) 🡪 (g) nằm trên các thread chạy song song với nhau. Hãy lập trình mô phỏng và đồng bộ trên C trong hệ điều hành Linux theo thứ tự sau:

* (c), (d) chỉ được thực hiện sau khi v được tính
* (e) chỉ được thực hiện sau khi w và y được tính
* (g) chỉ được thực hiện sau khi y và z được tính

Trả lời...