BÁO CÁO THỰC HÀNH

**Môn học: Hệ điều hành (IT007)**

**Lab 05: ĐỒNG BỘ HOÁ TIẾN TRÌNH, TIỂU TRÌNH**

*GVHD: Phạm Quốc Hùng*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: IT007.N25.1

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Lê Đoàn Trà My | 21521149 | 21521149@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | Câu 1: | Hoàn thành tốt, 100% |
| 2 | Câu 2: | Hoàn thành tốt, 100% |
| 3 | Câu 3: | Hoàn thành tốt, 100% |
| 4 | Câu 4: | Hoàn thành tốt, 100% |
| Điểm tự đánh giá | | 9.5 - 10 |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

**Câu 1:** Hiện thực hóa mô hình trong ví dụ 5.3.1.2, tuy nhiên thay bằng điều kiện sau: sells <= products <= sells + [2 số cuối của MSSV + 10]

A screenshot of a computer screen

Description automatically generated with medium confidence

*Hình 1. Code bài 1 và kết quả đầu khi thực thi file bai1.c*

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

*Hình 2,3. Kết quả chạy code tiếp theo (ở phần cuối kết quả chạy được)*

**Câu 2:** 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 screenshot of a computer program

Description automatically generated with medium confidence

*Hình 4. Code chưa đồng bộ hoá semaphore*

*A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated*

*Hình 5,6,7. Kết quả thực thi code chưa đồng bộ hoá semaphore*

- Nhận xét về lỗi khi chương trình chưa được đồng bộ: 2 process add\_element và remove\_element chạy song song, truy cập vào biến toàn cục n đồng thời có thể xảy ra tình huống đua nhau ghi vào biến n dẫn đến sai sót trong kết quả (Trong lúc thêm phần tử vào thì hàm remove\_element có thể thực hiện lấy phần tử ra khỏi mảng và ngược lại do đó kết quả không chính xác).

- Tiến hành đồng bộ hoá và thực thi lại:

A screenshot of a computer screen

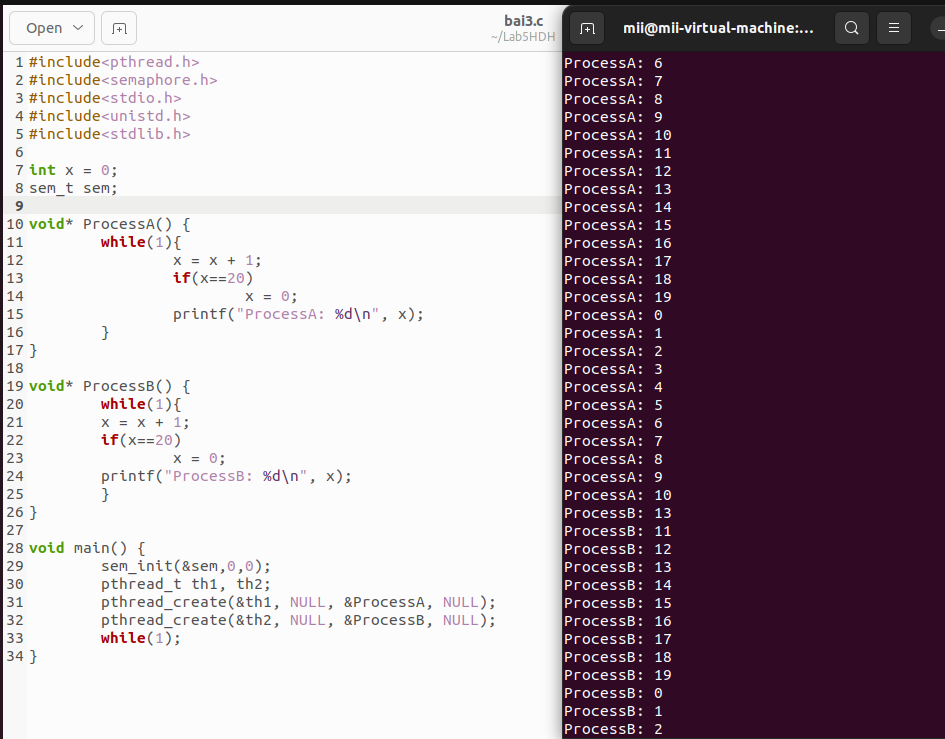
Description automatically generated with medium confidence

*Hình 8. Kết quả chạy code đồng bộ hoá semaphore*

**Câu 3:** Cho 2 process A và B chạy song song như sau:

|  |  |
| --- | --- |
| int x = 0; | |
| PROCESS A | PROCESS B |
| process A()  {  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.



*Hình 9. Kết quả chạy code bài 3*

- Nhận xét: Do 2 tiến trình chưa được cài đặt đồng bộ dẫn đến lỗi Mutual Section, xảy ra khi cả 2 tiến trình đều được cho vào vùng tranh chấp. Khi một trong hai tiến trình đang thực hiện các công việc bất kì thì giữa chừng bị tiến trình khác xen ngang vào làm kết quả bị sai.

**Câu 4:** Đồng bộ với mutex để sửa lỗi bất hợp lý trong kết quả của mô hình Bài 3.

**A screenshot of a computer screen

Description automatically generated with medium confidence**

*Hình 10. Kết quả chạy code bài 4*

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)