|  |
| --- |
| Name: Phạm Đức Thể  ID: 19522253  Class: IT007.M14.2 |

OPERATING SYSTEM  
LAB 5 REPORT

**SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | | **Status** | **Page** |
| Section 5.4 | Ex 1 | Hoàn thành | 2 |
| Ex 2 | Hoàn thành | 4 |
| Ex 3 | Hoàn thành | 8 |
| Ex 4 | Hoàn thành | 10 |
|  |  |  |
|  |  |  |
|  |  |  |
| Section 5.5 | Ex 1 | Hoàn thành | 12 |
|  |  |  |

**Self-scrores: 10/10**

*\*Note: Export file to* ***PDF*** *and name the file by following format:* ***LAB X – <Student ID>.pdf***

# Section 5.4

## Task name 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]

Text

Description automatically generated

Hình 1: Process thực hiện việc cộng Products được chạy trước.

Text

Description automatically generated

Hình 2: Sau khi đạt giới hạn, Products bị buộc dừng bởi sem\_wait và Sells được chạy.

Text

Description automatically generated

Hình 3: Source code bài 1

* **Giải thích**: sem1 đóng vai trò là điều kiện (sells<=products), sem2 đóng vai trò là điều kiện (products <= sells + 63). Khi chương trình được nạp, giả sử ProcessA được nạp trước. Khi đó A sẽ bị khóa lại với bởi biến sem1 (khởi tạo = 0). Vì vậy ProcessB sẽ được chạy. Products và sem1 (bởi hàm sem\_post) sẽ được tăng lên cho đến khi sem2=0 (được giảm bởi hàm sem\_wait). Khi đó ProcessA sẽ được chạy. Sells và sem2 (bởi hàm sem\_post) sẽ tăng cho đến khi sem1=0 (được giảm bởi hàm sem\_wait). Hai tiến trình này sẽ được lặp đi lặp lại trong 1 vòng lặp vô hạn.

## Task name 2: Chạy thử và tìm ra lỗi khi chạy chương trình 2 thread song song khi:

1. Chưa được đồng bộ:

Text

Description automatically generated

Hình 4: Tiến trình sinh ra phần tử và bỏ vào a được chạy trước.

Text

Description automatically generated

Hình 5: Lỗi xuất hiện khi chưa đồng bộ.

Text

Description automatically generated

Hình 6: Soucre code bài 2a

1. Đã được đồng bộ:

Text

Description automatically generated

Hình 7: Số phần tử trong a được thêm vào và lấy ra luôn có sự đồng nhất về dữ liệu.

Text

Description automatically generated

Hình 8: Tương tự khi 2 bên đổi vị trí.

A picture containing graphical user interface

Description automatically generated

Hình 9: Soucre code bài 2b

* **Giải thích**: Hàm arrange đóng vai trò sắp xếp lại hàm khi lấy ngẫu nhiên một phần tử trong a ra. Biến sem1 đóng vai trò điều kiện đảm bảo rằng sẽ không thể lấy phần tử trong A khi không có phần tử nào. Biến sem2 đóng vai trò điều kiện đảm bảo rằng sẽ không thể thêm phần tử quá số lượng phần tử được cho phép. Khi chương trình được nạp, giả sử ProcessB được nạp trước sẽ bị khóa bởi biến sem1 (khởi tạo = 0). Khi đó ProcessA được chạy, tiếp tục thêm vào phần tử có giá trị ngẫu nhiên vào a và tăng sem1 (bởi hàm sem\_post) cho đến khi sem2=0 (được giảm bởi hàm sem\_wait). Khi đó ProcessB được chạy, lấy một phần tử ngẫu nhiên trong a ra và tăng sem2 sem1 (bởi hàm sem\_post) cho đến khi sem1 = 0 (được giảm bởi hàm sem\_wait). Hai tiến trình này sẽ được lặp đi lặp lại trong 1 vòng lặp vô hạn.

## Task name 3: Hiện thực mô hình trong hệ điều hành Linux và nhận xét kết quả.

Text

Description automatically generated

Hình 10: Kêt quả chạy code bài 3

Lỗi xuất hiện khi chưa được đồng bộ. Process A đã lấy giá trị của biến x ở thời điểm Process B đang tính toán x=15, do đó sau khi Process B kết thúc quá trình chạy thì Process A lập tức in ra kết quả đã tính trước đó.

Text

Description automatically generated

Hình 11: Soucre code bài 3

## Task name 4: Đồng bộ với mutex để sửa lỗi bất hợp lý trong kết quả của mô hình Bài 3

Text

Description automatically generated

Hình 12: Process A kết thúc và lập tức Process B tiếp tục tính toán theo kết quả cuối cùng của Process A

Text

Description automatically generated

Hình 13: Tương tự khi Process B kết thúc.

Text

Description automatically generated

Hình 14: Soucre code bài 4

* **Giải thích**: biến mutex đóng vai trò là chìa khóa đóng mở vùng tranh chấp, tránh tình trạng process lấy giá trị của biến trong vùng tranh chấp khi 1 process đang chạy làm dữ liệu trở nên không đồng nhất.

# Section 5.5

## Task name 1: Lập trình mô phỏng và đồng bộ trên Ctrong hệ điều hành Linux chương trình theo yêu cầu đề bài:

Text

Description automatically generated

Hình 15: Kết quả thực thi source code bài 1

Process a) và b) được chạy trước.

Lần lượt các process c) d) e) g) được chạy.

* **Giải thích**: Các biến sem\_AB, sem\_CD, sem\_EF và sem\_G là các biến semaphore đóng vai trò điều kiện giúp các process khác được chạy. Các biến sem\_CD\_extra và sem\_EF\_extra đóng vai trò điều kiện lần lượt cho các process CD và EF được chạy. Biến busy đóng vai trò là mutex, khóa và mở vùng tranh chấp. Khi chương trình được nạp, Giả sử các Process khác AB được chạy trước:

**+** Process G sẽ bị khóa bởi biến sem\_EF (khởi tạo=0).

+ Process EF sẽ bị khóa bởi biến sem\_EF\_extra (khởi tạo=0).

+ Process CD sẽ bị khóa bởi biến sem\_CD\_extra (khởi tạo=0).

+ Process AB sẽ được chạy do biến semG được khởi tạo =1 và khóa vùng tranh chấp.

+ Sau khi AB chạy xong, các điều kiện sẽ được kích hoạt để CD có thể chạy (Các điều kiện này không xung đột với điều kiện ở các process còn lại), đồng thời mở vùng tranh chấp.

+ Sau khi CD chạy xong, tiếp tục tương tự với AB khi cho EF chạy tiếp.

+ EF chạy xong kích hoạt điều kiện cho G được chạy.

+ G sau khi chạy xong, kích hoạt điều kiện để cho A được chạy tiếp, tiếp tục chương trình như 1 vòng lặp.

A picture containing graphical user interface

Description automatically generated

Hình 16: Source code bài 1