FACULTY OF INFORMATION TECHNOLOGY
>>> COURSE MATERIAL <<<

INTRODUCTION TO OPERATING SYSTEM
Course ID 502047

Thông tin dưới đây được dịch từ tài liệu StudyGuide tại trang OS-Book của sách giáo trình chính và phần Summary của sách. Mọi lỗi sai sót hay góp ý, xin gửi về cho tôi qua email trantrungtin@tdtu.edu.vn

Study Guide for Lecture notes ch06

- Điều kiện cạnh tranh (race condition): vài tiến trình truy cập và thao tác lên biến số chia sẻ một cách đồng thời, kết quả có thể khác biệt tuỳ vào thứ tự các thao tác trên biến số.

- Mỗi tiến trình có một số đoạn mã nguy cơ (critical section), là các đoạn có thao tác lên dữ liệu chia sẻ.

+ Để giải quyết bài toán đoạn mã nguy cơ, mỗi tiến trình sẽ xin phép trước khi thực thi đoạn mã đó bằng đoạn lệnh đi vào (entry section), và sau khi kết thúc thực thi đoạn mã nguy cơ, tiến trình cần báo hiệu bằng đoạn lệnh rời khỏi (exit section) sau đó tiếp tục thực thi phần mã còn lại (remainder section), phần mà không có ảnh hưởng gì đến các biến số dùng chung.

+ Rất khó để giải quyết bài toán này trong các nhân tính toán có khả năng chiếm quyền ưu tiên.

- Peterson: giải pháp cho 2 tiến trình.

+ Các biến số chia sẻ giữa 2 tiến trình: int **turn** và Boolean **flag[2]**

+ **turn:** giá trị thể hiện cho tiến trình nào được mời vào đoạn mã nguy cơ.

+ **flag:** báo hiệu một tiến trình có nhu cầu thực thi đoạn mã nguy cơ tại thời điểm xét hay không.

▪flag[i] = true xác nhận rằng tiến trình P_i sẵn sàng chạy đoạn mã nguy cơ.

- Các máy tính hiện đại cung cấp các lệnh đơn nguyên phần cứng. Atomic = không thể bị ngắt.

- Giải pháp sử dụng khoá Locks:

```
do {
        acquire lock
             critical section
       release lock
             remainder section
    } while (TRUE);
```

- Giải pháp Thử và Thiết lập Test-And-Set: Các khoá luận lý chia sẻ, khởi tạo là FALSE.

- Giải pháp sử dụng Swap: Ngoài các khoá luận lý chia sẻ, khởi tạo là FALSE; mỗi tiến trình có khoá riêng.

- Semaphore: Công cụ đồng bộ mà không bị hạn chế bận chờ đợi.

+ Các thao tác đính kèm: wait() và signal() ← chỉ có các thao tác này mới có quyền truy cập semaphore S.

+ Semaphore có thể là kiểu số nguyên counting (dãy không giới hạn) hay kiểu nhị phân binary (0 hoặc 1).

- Tắc nghẽn (Deadlock): 2 hoặc nhiều hơn các tiến trình chờ đợi vô hạn định cho một sự kiện mà chỉ có thể xảy ra bởi một tiến trình đang chính trong các tiến trình chờ đợi đó. Phần lớn các hệ điều hành không phòng bị hay xử lý các tắc nghẽn.

+ Có thể gây ra cạn kiệt tài nguyên starvation và đảo lộn ưu tiê priority inversion (tiến trình ưu tiên thấp giữ các khoá mà tiến trình ưu tiên cao đang cần).

- Một số bài toán đồng bộ khác bao gồm Bounded-Buffer Problem và Readers-Writers Problem.

- Bộ quan sát (Monitor) là một trừu tượng mức cao, cung cấp cơ chế đồng bộ hiệu quả và tiện lợi.

+ Chỉ có duy nhất một tiến trình được hoạt động bên trong một bộ quan sát tại một thời điểm.

+ Chúng ta có thể tuỳ biến các biến số điều kiện (condition variables) để tạm dừng hay khởi chạy lại một tiến trình (ví dụ condition x, y);

▪ x.wait() – tiến trình đã thực hiện gọi sẽ bị tạm dừng cho đến khi có thực thi lệnh x.signal()

▪ x.signal() – khởi chạy tiếp tục một trong những tiến trình đang bị tạm dừng.

+ Có thể hiện thực thông qua semaphore.

Summary Chapter 06 of book "OS concepts 10th edition"

- Tình trạng cạnh tranh (race condition) xảy ra khi các tiến trình có quyền truy cập đồng thời vào dữ liệu chia sẻ và kết quả có thể khác nhau, phụ thuộc vào thứ tự thực thi cụ thể trong quá trình xảy ra truy cập đồng thời. Điều kiện cạnh tranh có thể dẫn đến các giá trị bị sai sót (không đồng nhất) của dữ liệu chia sẻ.

- Đoạn mã nguy cơ / Đoạn mã cạnh tranh là phần mã nguồn nơi dữ liệu chia sẻ có thể bị thao tác sai sót và tình trạng cạnh tranh có thể xảy ra. Bài toán đoạn mã nguy cơ là yêu cầu một thiết kế một giao thức, mà theo đó các tiến trình có thể đồng bộ hóa hoạt động của chúng để hợp tác cùng chia sẻ dữ liệu.

- Bất kỳ một giải pháp cho bài toán đoạn mã nguy cơ phải đáp ứng ba yêu cầu sau: (1) loại trừ lẫn nhau, (2) đảm bảo sự tiến triển và (3) chờ đợi có hạn định. Loại trừ lẫn nhau đảm bảo rằng chỉ có một tiến trình tại một thời điểm được thực thi đoạn mã nguy cơ của nó. Đảm bảo sự tiến triển cần đảm bảo rằng các tiến trình sẽ hợp tác xác định tiến trình nào tiếp theo sẽ thực thi đoạn mã nguy cơ của nó. Chờ đợi có hạn định sẽ đảm bảo thời gian một tiến trình chờ trước khi nó có thể thực thi đoạn mã nguy cơ của nó là có hạn định.

- Các giải pháp phần mềm cho vấn đề đoạn mã nguy cơ, ví dụ giải pháp Peterson, không hoạt động tốt trên các kiến trúc máy tính hiện đại.

- Hỗ trợ phần cứng cho vấn đề đoạn mã nguy cơ bao gồm các rào cản bộ nhớ (memory barriers); chỉ thị phần cứng (hardware instructions), ví dụ compare-and-swap(); và các biến số đơn nguyên.

- Khóa loại trừ lẫn nhau (Mutex lock) cung cấp loại trừ lẫn nhau bằng cách yêu cầu tiến trình có được khóa trước khi vào đoạn mã nguy cơ và giải phóng khóa khi thoát khỏi đoạn mã nguy cơ vừa thực thi xong.

- Semaphore, như khóa mutex, có thể được sử dụng để đảm bảo điều kiện loại trừ lẫn nhau. Tuy nhiên, trong khi khóa mutex sử dụng giá trị nhị phân để thể hiện khóa có khả dụng hay không, semaphore có giá trị nguyên và do đó có thể được sử dụng để giải quyết nhiều bài toán đồng bộ hóa khác.

- Bộ giám sát là mô hình trừu tượng cung cấp hình thức đồng bộ hóa tiến trình cấp cao. Một bộ quan sát sử dụng các biến điều kiện cho phép các tiến trình chờ đợi một số điều kiện cụ thể và báo hiệu cho nhau khi các điều kiện đó được đáp ứng.

- Giải pháp cho vấn đề đoạn mã nguy cơ có thể gặp phải các vấn đề về tính sống còn, bao gồm cả tắc nghẽn.

- Còn có nhiều công cụ khác có thể được sử dụng để giải quyết vấn đề đoạn mã nguy cơ cũng như để đồng bộ hóa hoạt động của các tiến trình và chúng có thể được đánh giá theo các mức độ tranh chấp khác nhau. Một số công cụ hoạt động tốt hơn dưới tải trọng tranh chấp nhất định so với những công cụ khác.