>>> 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 ch07

(Review examples of Synchronization).

Summary Chapter 07 of book "OS concepts 10th edition"

- Các bài toán kinh điển về đồng bộ hóa tiến trình bao gồm bộ đệm giới hạn (bounded-buffer), độc giả và nhà văn (readers–writers) và các nhà triết học ăn tối (dining-philosophers). Các giải pháp cho các bài toán này sử dụng các công cụ được trình bày trong Chương 6, bao gồm khóa mutex, semaphores, bộ giám sát và các biến điều kiện.

- Windows sử dụng các đối tượng điều phối cũng như các sự kiện để hiện thực các công cụ đồng bộ hóa tiến trình.

- Linux sử dụng nhiều cách tiếp cận khác nhau để bảo vệ chống lại các tình trạng cạnh tranh, bao gồm các biến số đơn nguyên, spinlocks và khóa mutex.

- API POSIX cung cấp các khóa mutex, semaphores và các biến điều kiện. POSIX cung cấp hai dạng semaphores: được đặt tên và vô danh. Một số tiến trình không liên quan có thể dễ dàng truy cập cùng một semaphore có tên bằng cách tham khảo tên của nó. Các semaphores vô danh không thể được chia sẻ một cách dễ dàng và chúng cần được đặt trong một vùng nhớ chia sẻ.

- Java có thư viện và API phong phú để đồng bộ hóa. Các công cụ có sẵn bao gồm các bộ quan sát (được cung cấp ở cấp độ ngôn ngữ) cũng như các khóa reentrant[1], semaphores và các biến điều kiện (được hỗ trợ bởi API).

- Các cách tiếp cận khác để giải quyết vấn đề đoạn mã nguy cơ bao gồm bộ nhớ giao dịch[2], OpenMP và ngôn ngữ chức năng[3]. Các ngôn ngữ chức năng đặc biệt hấp dẫn, vì chúng cung cấp một mô hình lập trình khác với ngôn ngữ thủ tục. Không giống như ngôn ngữ thủ tục, ngôn ngữ chức năng không duy trì trạng thái và do đó thường miễn dịch với các tình trạnh cạnh tranh và đoạn mã nguy cơ.

[1] https://en.wikipedia.org/wiki/Reentrant_mutex

[2] https://en.wikipedia.org/wiki/Transactional_memory

[3] https://en.wikipedia.org/wiki/Functional_programming