KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 9.1 – Tắc nghẽn.

Trong lập lịch CPU, các yếu tố được xem xét và Bộ lập lịch sẽ dựa vào độ ưu tiên, quy tắc xoay vòng hay FIFO để vừa đảm bảo hiệu năng vừa không mất tính công bằng cho các tiến trình. Trong LAB này, một số tình huống lập lịch được giới thiệu.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Tắc nghẽn trong Đa luồng	Ch 8.2 Deadlock in Multithreaded Applications	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
Ví dụ trylock		
Triết gia ăn tối		

Yêu cầu sinh viên: Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng cho các bài toán liên lạc giữa các tiến trình.

Đánh giá sinh viên: Hỏi đáp các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các Yêu cầu trong buổi thực hành và Bài tập cuối hướng dẫn trong thời gian cho phép của giảng viên.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

Programming Problems of Chapter 8.

[2] Wikipedia, [2021], Dining philosophers problem,

Access https://en.wikipedia.org/wiki/Dining_philosophers_problem

[3] Linux manual page, [2021], on pthread_mutex_lock(3p)

Access https://man7.org/linux/man-pages/man3/pthread_mutex_lock.3p.html

[4] Subham Biswas, [2021], Dining Philosopher Problem Using Semaphores,

Access https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/

Yêu cầu 1: Deadlock diễn ra với khoá Mutex như thế nào?

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/CHAPTER_8/Fig8_01_Deadlock.c***, biên dịch và thực thi. Lưu ý lệnh biên dịch cần thêm chọn lựa "tiểu trình".

- Dòng 14 và 15 là hai khoá mutex.

- Dòng 36 và 38 thể hiện yêu cầu khoá của tiểu trình thứ nhất.

- Dòng 48 và 50 thể hiện yêu cầu khoá của tiểu trình thứ hai.

```
14      pthread_mutex_t first_mutex;
15      pthread_mutex_t second_mutex;

36        pthread_mutex_lock( & first_mutex);
37         sleep(1);
38        pthread_mutex_lock( & second_mutex);
             // Do some work
42        pthread_mutex_unlock( & second_mutex);
43        pthread_mutex_unlock( & first_mutex);

48        pthread_mutex_lock( & second_mutex);
49         sleep(1);
50        pthread_mutex_lock( & first_mutex);
             // Do some work
53        pthread_mutex_unlock( & first_mutex);
54        pthread_mutex_unlock( & second_mutex);
```

?	- Các lệnh trả khoá 42, 43, 53 và 54 có được thực thi không? - Chương trình có kết thúc không?	Ch8 p.319	Slide Ch8.5-7

	- Nếu thay đổi thứ tự gọi khoá (đổi dòng 48 và 50) với nhau thì kết quả thực thi như thế nào?		

Yêu cầu 2: Deadlock có thể tránh với **try_lock**() như thế nào? Giải thích.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/CHAPTER_8/Fig8_02_trylock.c***, biên dịch và thực thi. Lưu ý lệnh biên dịch cần thêm chọn lựa "tiểu trình".

?	- Vì sao trylock() có thể gỡ được tắc nghẽn? - Các tiểu trình mặc dù không tắc nghẽn nhưng bị Starvation (thiếu tài nguyên) vì sao? - Tính liveness của một chương trình là gì?	Ch6 p.283	Slide Ch8.15

Yêu cầu 3: Hiện thực bài toán Triết gia ăn tối, đồng bộ bằng semaphore, chạy thử và giải thích khả năng tắc nghẽn của chương trình.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_9/task1_3_DP.c***, biên dịch và thực thi.

?	- Khi nào sẽ xảy ra tắc nghẽn? - Nếu áp dụng ý tưởng "trylock", tức là không lấy được chiếc đũa thứ hai thì phải trả lại cả chiếc đũa thứ nhất vừa lấy được, thì còn khả năng tắc nghẽn không?* - Tính liveness của một chương trình như thế nào nếu câu * phía trên được trả lời là "không".	Ch6 p.283	Slide Ch8.15

Bài tập lập trình.

- Nộp các tập tin đã biên dịch và thực thi.