KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 9.3 – Giải thuật Banker.

Trong hệ thống có nhiều loại tài nguyên, mỗi tài nguyên lại có nhiều thực thể. Các tiến trình sẽ khai báo số lượng thực thể mà chúng cần để thực thi và hoàn tất. Trong vai trò Phân phối và quản lý tài nguyên, Hệ điều hành cần đánh giá tính an toàn của hệ thống để có những phương án định thời và cấp phát phù hợp.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Giải thuật An toàn	Ch8.6.3 Banker's Algorithm Ch8.6.3.1 Safety Algorithm Ch8.6.3.2 Resource-Request Algorithm	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
Giải thuật Yêu cầu tài nguyên		
Giải thuật Banker		

Yêu cầu sinh viên: Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng cho các bài toán tìm chuỗi An toàn.

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] Vikash Kumar, [2020], Banker's Algorithm in Operating System, Geeksforgeeks.org.

Access https://www.geeksforgeeks.org/bankers-algorithm-in-operating-system-2/

Yêu cầu 1: Tìm chuỗi an toàn của Hệ thống cho trong Ví dụ minh hoạ 8.6.3.3 (sách [1] trang 336) (cũng là trong Slide bài giảng Ch8 slide 31).

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_9/task3_1_SafetyAlgorithm.c***, biên dịch và thực thi. Lưu ý có thể cần đặt lại các giá trị các ma trận cho khớp với thông số bài toán.

```
$ gcc -o banker.out task3_1_SafetyAlgorithm.c
$ ./banker.out 3 3 2
$ SAFE Sequence:  P1 -> P3 -> P4 -> P0 -> P2.

$ ./banker.out 3 2 2
$ The system is UNSAFE.
```

Yêu cầu 2: (SV tự học) (Resource-Request Algorithm) Hiện thực hàm

```
int request_resources(int customer_num, int request[]);
```

* Tham số: `customer_num` là chỉ số tiến trình yêu cầu thêm tài nguyên và mảng request[] chứa số thực thể mà tiến trình `customer_num` muốn có lập tức.

* Trả về: nếu việc cấp phát không tiến hành được, -1 được trả về; ngược lại 0 được trả về.

Hướng dẫn:

- Lý thuyết: sách [1] trang 336 và Slide bài giảng Ch8 slide 30.

- Mã giả: tại ***LAB_IntroOS/LAB_9/task3_2_Resource_Request_Pseudocode.c***

Yêu cầu 3: (SV tự học) Hiện thực hàm

```
void release_resources(int customer_num, int request[]);
```

* Tham số: `customer_num` là chỉ số tiến trình trả lại tài nguyên và mảng request[] chứa số thực thể mà tiến trình `customer_num` phải trả lại lập tức.

* Trả về: không.

Ứng dụng: trong một số trạng thái Tắc nghẽn, hoặc Cạn kiệt tài nguyên; Hệ thống sẽ lấy lại những thực thể mà một hoặc vài tiến trình đang giữ (Trên thực tế, Hệ thống sẽ kết liễu hoàn toàn tiến trình). Điều này làm cho tài nguyên available tăng lên và có thể giúp cho hệ thống tiếp tục thực thi.

Bài tập

1. Bổ sung mã vào Yêu cầu 1 để đọc tập tin và nhập vào hai ma trận maximum và allocation từ tập tin.
   Gợi ý: các tập tin mẫu được cho task3_maximum.txt và task3_allocation.txt. Hàm đọc tập tin tương tự trong LAB 4.3 – Yêu cầu 2 (mã nguồn ***LAB_IntroOS/LAB_4/task3_2.c***), trong đó, hàm đọc tập tin có thể dùng cho 2 tiểu trình để đọc 2 tập tin cùng lúc.

2. Tham số truyền vào trong lời gọi là vector available, nếu tham số này là "tổng số tài nguyên hệ thống có ban đầu" thì vector available được tính như thế nào, hãy sửa đổi đoạn mã trong Yêu cầu 1.
   Gợi ý: available là số thực thể tài nguyên còn lại sau khi hệ thống đã cấp phát cho các tiến trình. Và tổng số thực thể tài nguyên đã cấp phát của 1 loại tiến trình chính là tổng các phần tử trong cột đại diện của tài nguyên đó trong ma trận allocation.

   Lời gọi trong Yêu cầu 1 lúc này tương đương:

```
$ ./banker.out 10 5 7
$ SAFE Sequence:  P1 -> P3 -> P4 -> P0 -> P2.
```