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

Hướng dẫn Lab 6.4 – Giải thuật định thời SJF

FCFS là giải thuật khá đơn giản để triển khai, tuy vậy, hiệu ứng Hộ tống có thể gây nên sự chờ đợi của đám đông và vì thế thời gian chờ trung bình rất lâu, cũng như ảnh hưởng đến tính đáp ứng của hệ thống. SJF là cách để giúp các tiến trình ngắn được xử lý trước và được xem là giải thuật tối ưu.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Lập lịch CPU	Ch5 - Programming Projects Scheduling Algorithms	https://github.com/Trantin84/LAB_IntroOS Sử dụng image Ubuntu 16 / 18 Project "CPU Scheduler"
Giải thuật SJF		
Bài tập SJF		

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 5.

[2] Greg Gagne , [2019], GitHub OS-BOOK OSC10e, Westminster College, United States

Access https://github.com/greggagne/osc10e in September 2019.

Yêu cầu 1: Hiện thực giải thuật SJF với giả định mọi tiến trình đến tại thời điểm 0.

Hướng dẫn: Nếu mọi tiến trình đã đến khi Bộ định thời ra quyết định (T_arrival <= Time) thì việc đáp ứng được dựa trên sự sắp xếp theo Thời gian chạy ngắn nhất (Shortest Job First) theo đúng tên của giải thuật.

Hãy sao chép LAB 6.3 thành LAB 6.4 và chỉnh sửa/bổ sung các chi tiết sau:

Tạo mới tập tin SJF.c

	FCFS.c	SJF.c
9 12	`void * FCFS(void * param) {` `  //SORTING ARRIVAL TIME`	`void * SJF(void * param) {` `  //SORTING BURST TIME`

Chỉnh sửa tập tin driver.c

`pthread_create( & tid[0], NULL, FCFS, NULL);`

thành `pthread_create( & tid[0], NULL, SJF, NULL);`

* SV có thể tạo thành một tiểu trình mới `pthread_create( & tid[1], NULL, SJF, NULL);` để chạy cả 2 giải thuật, khi đó, cần sao chép mảng task cho từng tiểu trình, vì mảng task ban đầu đã bị thao tác thay đổi bởi mỗi tiểu trình được gọi. Cũng cần lưu ý khi chạy đồng thời nhiều giải thuật, các dòng in thông báo ra màn hình có thể bị xen kẽ.

Chỉnh sửa tập tin task.h

- Thêm các nguyên mẫu hàm vừa bổ sung nếu có.

- Thêm các biến số toàn cục nếu có.

```
$make clean
$make sjf
$./sjf data.txt
```

Yêu cầu 2: Hiện thực giải thuật SJF với giả định mọi tiến trình đến tại các thời điểm khác nhau và khác 0.

Hướng dẫn: Giải thuật SJF xem xét phục vụ Tiến trình ngắn hơn trước, nhưng chỉ những tiến trình đã và vừa đến lúc đó mới được xem xét. Để hiện thực hai yêu cầu này, chúng ta cần 2 giai đoạn:

1. Giai đoạn 1: Sắp xếp các tiến trình theo thứ tự đến
2. Giai đoan 2: Chọn tiến trình có thời gian chạy nhỏ nhất trong tập các tiến trình có thời điểm đến trước hoặc ngay thời điểm xét của bộ định thời. Phục vụ tiến trình này, loại bỏ nó ra khỏi danh sách. Và lặp lại bước 2 nếu danh sách chưa rỗng.
3. Trong trường hợp CPU đang rảnh mà các tiến trình vẫn chưa đến, bộ định thời sẽ tăng dần thời gian time (time++) cho đến khi có tiến trình đến. Đoạn thời gian này gọi là Idle time.

Bài tập

Nộp các phiên bản ở các Yêu cầu nêu trên cùng các tập tin dữ liệu.

Phụ lục

Bài tập 5.4 Xem tập các tiến trình sau đây, với thời gian cần chạy ở cột Burst Time được cho ở đơn vị mili giây.

Process	Burst Time	Priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

Cho rằng các tiến trình đến theo thứ tự P1, P2, P3, P4, P5, tại thời điểm 0.0

Bài tập 5.5 Các tiến trình sau đây được lập lịch với giải thuật RR

Process	Priority	Burst Time	Arrival
P1	40	20	0
P2	30	25	25
P3	30	25	30

P4	35	15	60
P5	5	10	100
P6	10	10	105