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

Hướng dẫn Lab 6.3 – Giải thuật định thời FCFS

FCFS là phương án định thời đơn giản và theo thói quen xếp hàng thường gặp tại các quầy phục vụ. FCFS mặc dù không tạo ra các tiêu chí tối ưu nhưng đơn giản để hiện thực nên luôn là một lựa chọn của các Hệ điều hành. Trong Lab này, chúng ta cùng xem qua cấu trúc dữ liệu cũng như các tác vụ phải hiện thực để định thời FCFS các tiến trình.

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 FCFS		
Bài tập FCFS		

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 4.

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

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

Yêu cầu 1: Hãy tạo các tập tin dữ liệu cho bài tập 5.4 và 5.5 cùng các ví dụ trong Slide bài giảng chương 5.

Hướng dẫn:

- Lịch trình của các tác vụ lưu trữ trong tập tin txt có định dạng trên mỗi dòng [tên tác vụ] [độ ưu tiên] [thời điểm đến] [CPU burst], với định dạng ví dụ sau (tải về từ ***LAB_IntroOS/LAB_6/task3_1data.txt***)

T1, 4, 0, 20

T2, 2, 0, 25

T3, 3, 0, 25

T4, 3, 0, 15

T5, 10, 0, 10

Trong đó, tác vụ T1 có mức độ ưu tiên 4 và chuỗi CPU là 20 đơn vị thời gian đến lúc 0, v.v.

Yêu cầu 2: tính thời gian đợi trung bình và thời gian quay vòng trung bình.

Hướng dẫn: Tải về tập tin được liệt kê dưới đây (trong ***LAB_IntroOS/LAB_6/task3_2/)***, biên dịch và thực thi.

task.h chứa danh sách thư viện cần thiết và định nghĩa cấu trúc dữ liệu các tác vụ.

driver.c có nhiệm vụ đọc tập tin dữ liệu, tạo thành mảng các tác vụ, tạo tiểu trình để thực thi các giải thuật được hiện thực.

FCFS.c Giải thuật FCFS được hiện thực trong tập tin này.

makefile Tập tin biên dịch cho thuận tiện.

Tập tin task.h

- Dòng 9 đến 15 là cấu trúc một tiến trình đọc từ tập tin dữ liệu.

- Dòng 19 đến 37 là các thư viện cần thiết.

- Dòng 41, 42 trở đi là các nguyên mẫu hàm, có thể bổ sung khi cần thiết.

```
9   typedef struct task {
10      char *name;
```

```
11      int tid;
12      int priority;
13      int burst;
14      int arrival;
15  } Task;

19-37 #include

41  void * FCFS(void * param);
42  void run(Task *task, int start, int slice);
```

Tập tin driver.c

- Dòng 19 tạo ra tiểu trình Đọc dữ liệu từ tập tin, đảm bảo quá trình đọc hoàn tất ở dòng 20, trước khi gọi các giải thuật.

- Dòng 23 tạo ra tiểu trình thực hiện giải thuật FCFS.

- Dòng 55, 59 là hiện thực chi tiết các hàm cần thiết.

```
19    pthread_create( & tid[0], NULL, reader, (argv[1]));
20    pthread_join(tid[0], & status);
23    pthread_create( & tid[0], NULL, FCFS, NULL);

55  void run(Task * task, int start, int slice)
59  void swap(Task * a, Task * b)
```

Tập tin FCFS.c

- Dòng 7 là số lượng tiến trình cần xử lý. Giá trị đã được xác định lúc hoàn thành đọc tập tin.

- Dòng 10 là biến số tổng các thời gian chờ và thời gian quay vòng.

- Dòng 11 là thời điểm bộ định thời thực thi.

- Dòng 12: trong yêu cầu này, chúng ta đã giả định thời điểm đến = 0 nên xem như mảng tiến trình đã được sắp xếp.

- Dòng 13 lặp lại đủ số lần để chạy tất cả tiến trình.

- Dòng 14 in thông tin ra màn hình.

- Dòng 15: Bộ định thời dịch đi một đoạn thời gian bằng thời gian chạy của tiến trình hiện tại.

- Dòng 16 là thời gian chờ của tiến trình hiện tại cộng dồn và tổng chung.

- Dòng 17: tương tự dòng 16 với thông tin là thời gian quay vòng.

```
7     extern int process;
10      int t_wait = 0, t_taround = 0;
11      int time = 0;
12      //SORTING ARRIVAL TIME
13      for (int i = 0; i < process; i++) {
14        run( & task[i], time, task[i].burst);
15        time += task[i].burst;
16        t_wait += time - task[i].burst - task[i].arrival;
17        t_taround += time - task[i].arrival;
```

Yêu cầu 3: sắp xếp các tiến trình nếu thời điểm đến (time arrival) của chúng khác nhau và khác 0.

Hướng dẫn: Trong Yêu cầu 2, chúng ta đã giả định rằng thời điểm đến của mọi tiến trình là 0, và phục vụ chúng theo thứ tự từ trên xuống trong tập tin dữ liệu, cũng là tuần từ từ đầu đến cuối mảng tiến trình. Trong Yêu cầu 3, thời điểm đến khác nhau, tức là tiến trình có thông tin thời điểm đến nhỏ hơn phải được mang ra phía trước mảng; mấu chốt của Yêu cầu này chính là sắp xếp mảng theo khoá "Thời điểm đến".

Hiện thực việc sắp xếp bằng hàm con hoặc trực tiếp tại dòng số 12 trong tập tin FCFS, sử dụng tập tin dữ liệu có thời điểm đến khác 0 và khác nhau để thử nghiệm.

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