Bài 4. LẬP LỊCH TIẾN TRÌNH

4.1 Mục tiêu

- Sinh viên nắm rõ được các giải thuật: First Come First Served (FCFS), Round Robbin (RR), Shortest Job First (SJF), Shortest Remain Time (SRT).
- Chỉ ra được ưu điểm và nhược điểm các giải thuật trên.
- Xây dựng được các chương trình mô phỏng các giải thuật trên.

4.2 Nội dung thực hành

- Mô phỏng giải thuật FCFS

4.3 Sinh viên chuẩn bị

Chương trước đã thảo luận về tiến trình – mức trừu tượng của Hệ điều hành đối với mã chương trình đang được thực thi. Chương này thảo luận về lập lịch tiến trình (hay còn gọi là lập lịch CPU) để tìm hiểu cách mà nhân Hệ điều hành cấp phát CPU cho tiến trình thực hiện công việc của nó.

Bộ lập lịch tiến trình sẽ quyết định tiến trình nào chạy, chạy khi nào và chạy trong bao lâu. Bộ lập lịch tiến trình sẽ chia tài nguyên processor time (tài nguyên hữu hạn) giữa các tiến trình đang chạy trong hệ thống. Bằng việc quyết định tiến trình nào sẽ

chạy tiếp theo, bộ lập lịch tiến trình có trách nhiệm sử dụng hệ thống tối ưu nhất và phải thể hiện được là nhiều tiến trình đang được thực thi đồng thời.

Hệ điều hành đa nhiệm có thể chạy xen kẽ nhiều tiến trình cùng một lúc. Trên các máy đơn bộ xử lý, các tiến trình sẽ được luân phiên sử dụng bộ xử lý trong một thời gian rất ngắn khiến cho người dùng có cảm giác nhiều tiến trình đang được chạy song song. Trên các máy nhiều bộ xử lý, nhiều tiến trình được thực sự chạy song song cùng nhau trên mỗi bộ xử lý. Dù là đơn bộ xử lý hay đa bộ xử lý, cũng sẽ có các tiến trình tồn tại trong bộ nhớ nhưng bị chặn (không được thực thi) cho đến khi nó được cấp phát processor time để chạy. Trong bài thực hành này, hệ thống đơn bộ xử lý được sử dụng để minh họa bộ lập lịch tiến trình.

Các bản phân phối Linux đều là hệ điều hành đa nhiệm ưu tiên. Trong các hệ điều hành đa nhiệm ưu tiên, bộ lập lịch tiến trình quyết định khi nào một tiến trình ngừng chạy và một tiến trình mới được bắt đầu chạy, việc tạm ngưng một tiến trình đang chạy được gọi là không tiếm quyền (tính ưu tiên). Thời gian mà một tiến trình chạy trước khi nó bị chặn là có thể dự đoán được và được gọi là timeslice của tiến trình.

Một tiến trình ở trạng thái đang chạy, nó có thể rời khỏi trạng thái bởi một trong ba lý do sau:

- Tiến trình đã hoàn thành công việc, khi đó nó trả lại processor time và chuyển sang chờ xử lý kết thúc.
- Tiến trình tạm dừng: Khi tiến trình chờ đợi một sự kiện nào đó, tiến trình sẽ được chuyển sang trạng thái thực hiện khi có xuất hiện sự kiện nó đang chờ.
- Tiến trình sử dụng hết processor time dành cho nó, khi đó sẽ được chuyển sang trạng thái chờ đến lượt cấp phát tiếp theo.

Việc chuyển tiến trình sang trạng thái chờ về bản chất là thực hiện việc phân phối lại processor time. Để điều khiển tiến trình ở nhiều trạng thái khác nhau, hệ thống thường tổ chức các từ trạng thái (thực chất là các khối điều khiển tiến trình) để ghi nhận tình trạng sử dụng tài nguyên và trạng thái tiến trình. Như vậy, lập lịch tiến trình có nghĩa là tổ chức một hàng đợi các tiến trình sẵn sàng để phân phối processor time cho chúng trên độ ưu tiên của các tiến trình; sao cho hiệu suất sử dụng bộ xử lý là tối ưu nhất. Mỗi tiến trình ở trạng thái sẵn sàng sẽ được gắn với một thứ tự ưu tiên. Thứ tự ưu tiên này được xác định dựa vào các yếu tố như: thời điểm hình thành tiến trình, thời gian thực hiện tiến trình, thời gian kết thúc tiến trình.

4.3.1 Giải thuật First Come First Served (FCFS)

Trong thuật toán này, độ ưu tiên phục vụ tiến trình căn cứ vào thời điểm hình thành tiến trình. Hàng đợi các tiến trình được tổ chức theo kiểu FIFO (vào trước, ra trước). Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt. Ưu điểm của thuật toán này là processor time không bị phân phối lại (không bị ngắt) và chi phí thực hiện thấp nhất (vì không phải thay đổi thứ tự ưu tiên phục vụ, thứ tự ưu tiên là thứ tự của tiến trình trong hàng đợi). Nhược điểm của thuật toán là thời gian trung bình chờ phục vụ của các tiến trình là như nhau (không kể thời gian tiến trình chạy ngắn hay dài), do đó dẫn tới ba điểm sau:

Ví dụ:

Tiến trình	**Thời điểm vào**	**Thời gian thực hiện**
P1	0	24
P2	1	3
P3	2	3

Thứ tự cấp phát processor time cho các tiến trình:

Tiến trình	**P1**	**P2**	**P3**
Thời điểm	0	24	27

Thời gian chờ trung bình: (0+23+25)/3=16.

4.3.2 Giải thuật Round robin (RR)

Giải thuật định thời luân phiên (round-robin scheduling algorithm - RR) được thiết kế đặc biệt cho hệ thống chia sẻ thời gian. Tương tự như định thời FCFS nhưng không còn tình trạng độc quyền processor time mà thay vào đó là các tiến trình sẽ lần lượt được cấp phát processor time với một định mức cố định, hay còn gọi là timeslice. Timeslice thường từ 10 đến 100 mili giây. Hàng đợi sẵn sàng được xem như một hàng đợi vòng. Bộ lập lịch tiến trình di chuyển vòng quanh hàng đợi sẵn sàng, cấp phát processor time tới mỗi tiến trình có khoảng thời gian tối đa bằng một timeslice. Để cài đặt định thời RR, chúng ta quản lý hàng đợi sẵn sàng như một hàng đợi FIFO của các tiến trình. Các tiến trình mới được thêm vào đuôi hàng đợi. Bộ lập lịch tiến trình chọn tiến trình đầu tiên từ hàng đợi sẵn sàng, đặt bộ đếm thời gian để ngắt sau 1 timeslice và gửi tới tiến trình. Sau đó, một trong hai trường hợp sẽ xảy ra. Tiến trình có processor time ít hơn 1 timeslice. Trong trường hợp này, tiến trình sẽ tự giải phóng. Sau đó, bộ lập lịch tiến trình sẽ xử lý tiến trình tiếp theo trong hàng đợi sẵn sàng. Ngược lại, nếu processor time của tiến trình đang chạy dài hơn 1 timeslice thì độ đếm thời gian sẽ báo và gây ra một ngắt tới hệ điều hành. Chuyển đổi ngữ cảnh sẽ được thực thi và tiến trình được đặt trở lại tại đuôi của hàng đợi sẵn sàng. Sau đó, bộ lập lịch tiến trình sẽ chọn tiến trình tiếp theo trong hàng đợi sẵn sàng.

- Ưu điểm:
  - Các tiến trình sẽ được luân phiên cho processor xử lý nên thời gian chờ đợi sẽ ít.
  - Đối với các tiến trình liên quan đến nhập/xuất, người dùng thì rất hiệu quả.
  - Việc cài đặt không quá phức tạp.
- Nhược điểm:
  - Thời gian chờ đợi trung bình theo RR thường là quá dài.
  - Nếu timeslice quá lớn thì RR thành FIFO.
  - Nếu timeslice quá ngắn so với thời gian xử lý của một tiến trình trong danh sách hàng đợi thì việc chờ đợi và xử lý luân phiên sẽ nhiều.
  - Quy tắc là timeslice nên dài hơn 80% processor time.

Ví dụ:

Tiến trình	**Thời điểm vào**	**Thời gian thực hiện**
P1	0	24
P2	1	3
P3	2	3

timeslice = 4

Thứ tự cấp processor time cho các tiến trình lần lượt là:

Tiến trình	**P1**	**P2**	**P3**	**P1**	**P1**	**P1**	**P1**	**P1**

Thời điểm	0	4	7	10	14	18	22	26

Vậy thời gian chờ đợi trung bình sẽ là: (3+5+6)/3 = 4,67. Như vậy RR có thời gian chờ đợi trung bình nhỏ hơn so với FIFO.

4.3.3 Giải thuật Shortest Job First (SJF)

Một tiếp cận khác đối với việc lập lịch tiến trình là giải thuật định thời công việc ngắn nhất trước (shortest job first - SJF). Khi bộ xử lý rảnh, nó được gán tới tiến trình có processor time kế tiếp ngắn nhất. Nếu hai tiến trình có cùng processor time kế tiếp, định thời FCFS sẽ được dùng.

- Ưu điểm:
  - ❖ Giải thuật được xem là tối ưu, thời gian chờ đợi trung bình giảm.
  - ❖ Tận dụng hết năng lực của bộ xử lý.
- Nhược điểm:
  - ❖ Cài đặt thuật toán phức tạp, tốn nhiều xử lý cho tiến trình quản lý.
  - ❖ Mặc dù SJF là tối ưu nhưng nó không thể được cài đặt tại cấp lập lịch tiến trình ngắn vì không có cách nào để biết chiều dài processor time tiếp theo.
  - ❖ Giải thuật SJF có thể độc quyền hay không độc quyền bộ xử lý, dẫn tới giải thuật này có nhiều

phiên bản khác nhau và sẽ tối ưu hay không tối ưu phụ thuộc vào chiến lược độc quyền bộ xử lý.

4.3.4 Giải thuật Shortest Remain Time (SRT)

Tương tự như SJF nhưng trong thuật toán này, độ ưu tiên thực hiện các tiến trình dựa vào thời gian cần thiết để thực hiện xong tiến trình (bằng tổng thời gian trừ đi thời gian đã thực hiện). Như vậy, trong thuật toán này cần phải thường xuyên cập nhật thông tin về thời gian đã thực hiện của tiến trình. Đồng thời, chế độ phân bổ lại processor time cũng phải được áp dụng nếu không sẽ làm mất tính ưu việt của thuật toán.

- Ưu điểm:
  - Thời gian chờ, tồn tại trong hệ thống của mỗi tiến trình đều ngắn.
  - Các tiến trình ngắn được thực thi nhanh chóng. Hệ thống cần ít chi phí cho việc ra quyết định tiến trình nào sẽ được thực thi tiếp theo.
- Nhược điểm:
  - Việc cài đặt thuật toán khá phức tạp.
  - Cần quản lý chặt chẽ việc điều phối các tiến trình.
  - Quản lý thời gian còn lại cho mỗi tiến trình.

4.3.5 Câu hỏi chuẩn bị

1. Vẽ sơ đồ giải thuật của các giải thuật lập lịch tiến trình:

- ❖ FCFS (First Come First Served)
- ❖ RR (Round Robin)
- ❖ SJF (Shortest Job First)
- ❖ SRT (Shortest Remain Time)

2. Giải thích các thuật ngữ sau:

TT	Thuật ngữ	Mô tả
1	Arrival time	
2	Burst time	
3	Quantum time (timeslice)	
4	Response time	
5	Waiting time	
6	Turnaround time	
7	Average waiting time	
8	Average turnaround time	

4.4 Hướng dẫn thực hành

Soạn thảo và biên dịch giải thuật FCFS bên dưới:

```
/*########################################
# University of Information Technology      #
# IT007 Operating System                    #
```

```
# <Your name>, <your Student ID>                #
# File: fcfs.c                                  #
########################################*/

#include<stdio.h>

void main(){
  int pn[10];
  int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], i, n;
  int totwt=0, tottat=0;
  printf("Enter the number of processes:");
  scanf("%d",&n);
  for(i=0;i<n;i++) {
    printf("Enter the Process Name, Arrival Time & Burst Time:");
    scanf("%d%d%d",&pn[i],&arr[i],&bur[i]);
  }
  for(i=0;i<n;i++) {
    if(i==0) {
      star[i]=arr[i];
      wt[i]=star[i]-arr[i];
      finish[i]=star[i]+bur[i];
      tat[i]=finish[i]-arr[i];
    } else{
```

```
      star[i]=finish[i-1];
      wt[i]=star[i]-arr[i];
      finish[i]=star[i]+bur[i];
      tat[i]=finish[i]-arr[i];
    }
  }
  printf("\nPName Arrtime Burtime Start TAT Finish");
  for(i=0;i<n;i++) {
printf("\n%d\t%6d\t\t%6d\t%6d\t%6d\t%6d",pn[i],arr[i],bur[i],star
[i],tat[i],finish[i]);
    totwt+=wt[i];
    tottat+=tat[i];
  }
}
```

Thực thi chương trình trên với đầu vào như sau:

- Số lượng tiến trình: 4
- Tiến trình đầu tiên (tên tiến trình, thời gian đến, thời gian xử lý): 0 0 2
- Tiến trình thứ hai: 1 1 3
- Tiến trình thứ ba: 2 2 4

Kiểm tra kết quả của chương trình có giống như bên dưới hay không?

PName	Arrtime	Burtime	Start	TAT	Finish
0	0	2	0	2	2
1	1	3	2	4	5
2	2	4	5	7	9

Tự kiểm tra kết quả của chương trình và gỡ lỗi nếu cần thiết. Sau đó bổ sung code để tính average waiting time và average turnaround.

4.5 Bài tập ôn tập

1. Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:
    - ❖ Nhập số lượng process
    - ❖ Nhập process name, arrival time, burst time
    - ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

2. Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:
    - ❖ Nhập số lượng process
    - ❖ Nhập process name, arrival time, burst time

- ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

3. Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):
    - ❖ Nhập số process
    - ❖ Nhập quantum time
    - ❖ Nhập process name, burst time
    - ❖ In ra Gantt chart với các thông số: process name, start processor time, stop processor time
    - ❖ In ra average waiting time và average turnaround time