Tổng hợp kiến thức

1, Khái niệm về thuật toán FCFS (First Come First Serve)

* Công việc nào tới trước sẽ xử lí trước
* Là thuật toán lập lịch CPU, thực hiện các yêu cầu và quy trình được xếp theo hàng đợi theo thứ tự đến của chúng

2, Đặc điểm

* Là thuật toán lập lịch CPU dễ dàng và đơn giản nhất
* Hàng đợi của tiến trình được tổ chức theo kiểu FIFO (First In First Out)
* Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt
* Khi tiến trình đi vào hàng đợi sẵn sàng, PCB (Process Control Block) của tiến trình đó dẽ được liên kết vào cuối hàng đợi theo nguyên tắc FIFO
  + Hàng đợi sẵn sàng: là nơi mà các tiến trình đang chờ được CPU xử lý. Khi một tiến trình được tạo ra và sẵn sàng để thực hiện, nó sẽ được đưa vào hàng đợi sẵn sàng
  + PCB (Process Control Block – Khối điều khiển tiến trình): là một cấu trúc quan trọng trong hệ điều hành, chứa tất cả các thông tin cần thiết để quản lý và theo dõi tiến trình đó; có thể hiểu đơn giản PCB giống như “hồ sơ” của một tiến trình
* Khi CPU hoàn thành việc xử lý một tiến trình và trở nên rảnh rỗi, nó sẽ lấy tiến trình đầu tiên của hàng đợi để thực hiện xử lý tiếp theo và điều này diễn ra tuần tự

3, Cách thức hoạt động của thuật toán

Thuật toán lập lịch trình FCFS thường được thông qua các số liệu được thu thập từ trước lập thành bảng dữ liệu và từ đó biểu diễn thành biểu đồ Gantt

* Ví dụ: Trong lịch trình này, ta có các sự kiện lần lượt là EV1, EV2, EV3, EV4 và EV5. Mỗi sự kiện sẽ có các thông số về Thời gian đến (Arrival time) và Thời gian bùng nổ (Burst time) tương ứng. Từ đó ta tính ra Thời gian hoàn thành (Completion time) và sau cùng tính Thời gian chờ (Waiting time) hay thêm Thời gian chờ trung bình (nếu bài toán yêu cầu)

|  |  |  |  |
| --- | --- | --- | --- |
| Sự kiện | Thời gian đến | Thời gian bùng nổ | Thời gian chờ |
| EV1 | 0 | 2 | 0 |
| EV2 | 3 | 4 | 8 |
| EV3 | 1 | 6 | 1 |
| EV4 | 6 | 5 | 9 |
| EV5 | 2 | 3 | 6 |

* ﻿﻿Các bước làm:
* ﻿﻿B1: Sắp xếp các sự kiện có Thời gian đến theo thứ tự từ bé đến lớn, sự kiện nào đến trước được thực thi trước
* ﻿﻿B2: Tính Thời gian hoàn thành theo phương thức sự kiện sau bằng tổng Thời gian bùng nổ của chính nó và các sự kiện trước
* ﻿﻿B3: Lập biểu đồ Gantt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EV1 | EV3 | EV5 | EV2 | EV4 |

0 2 8 11 15 20

* B4: Tính Thời gian chờ theo công thức:  
  Thời gian chờ = Thời gian bắt dầu - Thời gian đến
* ﻿﻿B5: Tính Thời gian chờ trung bình (nếu có)  
  Thời gian chờ trung bình = (0 +8 +1 +9 +6) :5 = 4,8

4, Lưu đồ thuật toán

A diagram of a diagram

Description automatically generated

• Chú thích:

* Hàng đợi: lưu trữ các lịch trình cần xử lí, mỗi lịch trình có các thông tin riêng như tên, thời gian đến, thời gian xử lí,...
* Thêm lịch trình mới: lịch trình mới sẽ được xếp vào cuối hàng đợi
* Thực hiện công việc: xử lí lịch trình đến trước đã lấy, các lịch trình khác phải chờ
* Xóa công việc: xóa lịch trình đã hoàn thành

5, Các bước xây dựng thuật toán FCFS

Các bước thực hiện thuật toán FCFS

**Bước 1: Khởi tạo danh sách các tiến trình**

* Tập hợp các tiến trình cần xử lý, mỗi tiến trình có thông tin về:
* Thời gian đến (Arrival Time)
* Thời gian thực thi (Burst Time)
* (Có thể thêm các thông tin khác như ID tiến trình)

Ví dụ: danh sách các tiến trình có thể như sau:

|  |  |  |
| --- | --- | --- |
| **Tiến trình** | **Thời gian đến (Arrival Time)** | **Thời gian thực thi (Burst Time)** |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 8 |
| P4 | 3 | 6 |

**Bước 2: Sắp xếp các tiến trình theo thời gian đến**

* Sắp xếp các tiến trình theo thứ tự thời gian đến (Arrival Time) từ nhỏ đến lớn.
* Nếu hai tiến trình có thời gian đến bằng nhau, thì chọn theo thứ tự mà chúng xuất hiện (ưu tiên tiến trình nào xuất hiện trước).

Ví dụ: trong bảng trên, danh sách đã được sắp xếp theo thời gian đến.

**Bước 3: Chọn tiến trình đầu tiên trong danh sách**

* Chọn tiến trình có thời gian đến sớm nhất từ danh sách.
* Tiến trình đó sẽ được thực thi đầu tiên.

Ví dụ: tiến trình P1 có thời gian đến là 0, sẽ được thực thi đầu tiên.

**Bước 4: Thực thi tiến trình**

* Tiến trình được chọn sẽ chạy liên tục cho đến khi hoàn thành mà không bị gián đoạn (FCFS là một thuật toán non-preemptive, tức là không giành quyền xử lý từ tiến trình khác).
* Cập nhật thời gian hoàn thành của tiến trình, thời gian chờ và thời gian quay vòng.

Ví dụ: tiến trình P1 được thực thi trong 5 đơn vị thời gian (từ thời điểm 0 đến 5).

**Bước 5: Cập nhật hàng đợi**

* Sau khi tiến trình vừa thực thi xong, loại bỏ tiến trình đó khỏi hàng đợi và chọn tiến trình tiếp theo trong danh sách dựa trên thời gian đến.
* Lặp lại quy trình này cho các tiến trình còn lại.

Ví dụ: sau khi P1 kết thúc ở thời điểm 5, ta sẽ chọn P2 vì nó có thời gian đến là 1 (nhỏ nhất trong số các tiến trình còn lại).

**Bước 6: Tính toán các thông số**

* Sau khi tất cả các tiến trình đã hoàn thành, tính toán các thông số hiệu suất:
* Thời gian chờ (Waiting Time): Thời gian một tiến trình phải chờ trước khi được thực thi:

Waiting Time = Start Time - Arrival Time

* Thời gian quay vòng (Turnaround Time): Tổng thời gian từ khi tiến trình đến cho đến khi nó hoàn thành.

Turnaround Time = Completion Time - Arrival Time

**Bước 7: Kết thúc**

* Sau khi tất cả các tiến trình đã được thực thi và danh sách tiến trình rỗng, thuật toán FCFS hoàn tất.

**Ví dụ thực hiện:**

Sử dụng ví dụ ở trên, giả sử chúng ta có các tiến trình P1, P2, P3, P4 với thời gian đến và thời gian thực thi như sau:

|  |  |  |
| --- | --- | --- |
| **Tiến trình** | **Thời gian đến** | **Thời gian thực thi** |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 8 |
| P4 | 3 | 6 |

**Tiến trình thực hiện:**

* Tại thời điểm 0, P1 được chọn và thực thi đến thời điểm 5.
* Tại thời điểm 5, P2 được chọn và thực thi đến thời điểm 8.
* Tại thời điểm 8, P3 được chọn và thực thi đến thời điểm 16.
* Tại thời điểm 16, P4 được chọn và thực thi đến thời điểm 22.

**Bảng tóm tắt kết quả:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tiến trình** | **Thời gian đến** | **Thời gian thực thi** | **Thời gian hoàn thành** | **Thời gian chờ** | **Thời gian quay vòng** |
| P1 | 0 | 5 | 5 | 0 | 5 |
| P2 | 1 | 3 | 8 | 4 | 7 |
| P3 | 2 | 8 | 16 | 6 | 14 |
| P4 | 3 | 6 | 22 | 13 | 19 |

**Kết quả:**

* Thời gian chờ trung bình = (0 + 4 + 6 + 13) / 4 = 5.75
* Thời gian quay vòng trung bình = (5 + 7 + 14 + 19) / 4 = 11.25

Trên đây là các bước thực hiện thuật toán FCFS.

6, Ưu điểm và nhược điểm của thuật toán

* Ưu điểm:
  + Hỗ trợ các loại lập lịch không ưu tiên lẫn ưu tiên
  + Công việc luôn thực thi theo thứ tự đến
  + Dễ sử dụng, đơn giản trong triển khai
* Nhược điểm:
  + Hiệu suất kém, thời gian đợi chung khá dài nếu một tiến trình có thời gian xử lí lâu (Hiệu ứng hộ tống – Convoy Effect)
  + Thiếu tính linh hoạt và không có khả năng ưu tiên

Hiệu ứng hộ tống: xảy ra khi một tiến trình lớn có thời gian thực thi dài chặn các tiến trình nhỏ hơn, dẫn đến hệ thống hoạt động kém hiệu quả

\*Tác động của hiệu ứng hộ tống

* Tăng thời gian chờ
* Giảm hiệu suất
* Hiệu ứng nghẽn cổ chai

\*Cách khắc phục

* Sử dụng các thuật toán lập lịch ưu tiên: SJF (Short Job First) hoặc RR (Round Robin)
* Lập lịch đa cấp độ

7, Ý tưởng thuật toán sắp xếp lịch trình

1. Thu thông tin sự kiện
2. Ứng dụng cho phép người dùng nhập các sự kiện với các thông tin gồm tên, thời gian bắt đầu, thời gian kết thúc thông qua giao diện (sheduleApp.py, interface.py).
3. Khi một sự kiện được thêm, nó sẽ được lưu trữ dưới dạng Event trong danh sách events.
4. Sắp xếp sự kiện kiện theo thời gian bắt đầu (FCFS)
5. Sử dụng thuật toán First Come First Served như trong FCFS.py để sắp xếp danh sách các sự kiện. Các sự kiện sẽ được sắp xếp theo thứ tự thời gian bắt đầu từ sớm đến muộn nhất.
6. Việc sắp xếp có thể thực hiện mỗi khi khi thêm 1 sự kiện mới hoặc kiểm tra nhắc nhở, đảm bảo danh sách luôn được sắp xếp đúng.
7. Kiểm tra và nhắc nhở sự kiện (sheduleApp.py, interface.py)
8. Chương trình sẽ chạy 1 luồng kiểm tra (check\_reminders), hoạt động song song và liên tục kiểm tra thời gian hiện tại so với các sự kiện trong danh sách.
9. Nếu thời gian hiện tại trùng với khoản thời gian và chưa gửi thông báo, thông báo sẽ được gửi đi. Sau đó, sự kiện được đánh dấu là đã thông báo để không bị nhắc lại.
10. Khi thời gian hiện tại vượt quá thời gian kết thúc sự kiện, chương trình sẽ gửi thông báo hoàn thành và loại bỏ sự kiện khỏi danh sách.
11. Lọc và hiển thị sự kiện (interface.py)
12. Ứng dụng hỗ trợ tìm kiếm sự kiện theo từ khóa. Khi người dùng nhập từ khóa, chương trình sẽ lọc các sự kiện có tên phù hợp và hiển thị trong cửa sổ danh sách.
13. Người dùng có thể mở cửa sổ phụ để xem chi tiết danh sách sự kiện, giúp quản lý lịch trình thuận tiện hơn.
14. Quản lý giao diện khay hệ thống (sheduleApp.py, interface.py)
15. Khi người dùng đóng cửa sổ chính, ứng dụng sẽ chuyển về chế độ ẩn và thêm icon vào khay hệ thống.
16. Từ khay hệ thống, người dùng có thể mở lại ứng dụng hoặc chọn thoát hoàn toàn. Menu khay cung cấp các tùy chọn này để dễ dàng truy cập và quản lý ứng dụng khi cần.
17. Cải tiến thêm
18. Tối ưu hóa cho nhiều sự kiện: Khi danh sách sự kiện trở nên lớn, thay vì FCFS, có thể triển khai các thuật toán tối ưu hơn như sắp xếp dựa trên loại sự kiện hoặc thời gian chờ. (FCFS.py)
19. Thông báo động: Để giảm tải cho chương trình, thay vì kiểm tra mỗi 10 giây (time.sleep(10)), có thể tính toán thời gian chờ tới sự kiện tiếp theo và chỉ kiểm tra khi cần. (scheduleApp.py, interface.py)
20. Tự động tái sắp xếp sau khi thêm sự kiện: Mỗi khi thêm sự kiện mới, ứng dụng sẽ tự động gọi hàm sắp xếp để cập nhật danh sách, đảm