**TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI**

**TP HỒ CHÍ MINH**

****

BÁO CÁO BÀI TẬP LỚN

**MÔN: HỆ ĐIỀU HÀNH**

**LẬP TRÌNH CÁC GIẢI THUẬT FCFS BẰNG NGÔN NGỮ**

**LẬP TRÌNH C++**

**Giáo viên hướng dẫn: Võ Tấn Sang**

**Sinh viên thực hiện:**

1. **Lê Vũ Trường 22H1120088**
2. **Nguyễn Tâm Hy 22H1120072**
3. **Hoàng Thiện 22H1120081**
4. **Phan Bảo Thành 22H1120080**
5. **Đào Văn Tuấn 22h1120089**

# Mục lục

[Mục lục 2](#_Toc164022155)

[Chương I. Giới thiệu về lập lịch CPU 3](#_Toc164022156)

[*1.1 Giới thiệu về lập lịch* 3](#_Toc164022157)

[*1.2 Khi nào nên lên lịch* 3](#_Toc164022158)

[*1.3 Thuật toán lập lịch không ưu tiên và lập lịch ưu tiên* 3](#_Toc164022159)

[**Chương II. Tổng quan về giải thuật FCFS** 3](#_Toc164022160)

[*2.1 Khái niệm* 4](#_Toc164022161)

[*2.2 Mô tả thuật toán* 4](#_Toc164022162)

[*2.3 Nguyên tắc hoạt động* 4](#_Toc164022163)

[*2.4 Ưu điểm và nhược điểm* 4](#_Toc164022164)

[*2.5 So sánh giải thuật FCFS với các giải thuật lập trình CPU khác (như SJF)* 4](#_Toc164022165)

[*2.6 Kết luận* 5](#_Toc164022166)

[**Chương 3. Cài đặt giải thuật FCFS bằng C++** 6](#_Toc164022167)

[*3.1 Các bước cài đặt thuật toán FCFS* 6](#_Toc164022168)

[*3.2 Code* 7](#_Toc164022169)

[*3.3 Kiểm thử* 12](#_Toc164022170)

# Chương I. Giới thiệu về lập lịch CPU

*1.1 Giới thiệu về lập lịch*

Lập lịch trong hệ điều hành là một quá trình quan trọng quyết định thứ tự và thời điểm các tiến trình được CPU xử lý.

Một trong những thuật toán lập lịch đầu tiên và cơ bản nhất được sử dụng trong các hệ điều hành là First-Come, First-Served (FCFS).

## *1.2 Khi nào nên lên lịch*

     Chìa khóa của việc lập lịch: các quyết định về việc lập trình.

     Các tình huống cần lập lịch:

* Tạo quy trình: Cần phải đưa ra quyết định về việc chạy tiến trình cha hay tiến trình con.
* Chấm dứt tiến trình: Một quyết định phải được đưa ra khi một tiến trình kết thúc. Quá trình đó không thể chạy được nữa, do đó một số tiến trình khác phải được chọn từ tập hợp các tiến trình sẵn sàng. Nếu không có tiến trình nào sẵn sàng, tiến trình rảnh rỗi do hệ thống cung cấp sẽ thường được chạy.
* Chặn tiến trình: Khi một tiến trình bị chặn, một tiến trình khác phải được chọn để chạy.
* Xảy ra gián đoạn do thiết bị IO hoặc phần cứng.

## *1.3 Thuật toán lập lịch không ưu tiên và lập lịch ưu tiên*

     Thuật toán lập lịch không ưu tiên:

* Chọn một tiến trình để chạy và sau đó cứ để nó chạy cho đến khi nó chặn hoặc cho đến khi nó tự nguyện giải phóng CPU (sẽ không bị treo cưỡng bức (forcibly suspended), không có quyết định lập lịch).
* Khi một tiến trình ở trạng thái chạy, nó sẽ tiếp tục cho đến khi chấm dứt hoặc tự chặn I/O.
* Áp dụng cho hệ thống batch.

     Thuật toán lập lịch ưu tiên:

* Tiến trình có thể chạy (liên tục) trong một khoảng thời gian cố định tối đa. Nếu nó chuyển sang trạng thái treo, bộ lập lịch sẽ chọn một quy trình khác để chạy ( cần có bộ hẹn giờ).
* Tiến trình đang chạy có thể bị gián đoạn và chuyển sang trạng thái sẵn sàng của hệ điều hành.
* Cho phép dịch vụ tốt hơn vì bất kỳ một tiến trình nào cũng không thể độc quyền (sử dụng) bộ xử lý trong thời gian rất dài.
* Áp dụng cho việc chia sẻ thời gian hoặc thời gian thực (real time).

**Chương II. Tổng quan về giải thuật FCFS**

*2.1 Khái niệm*

FCFS là viết tắt của First-Come-First-Served, công việc nào tới trước sẽ được xử lý trước. Ví dụ chúng ta đang xếp hàng để đợi đến lượt mua vé xem phim, ai đến trước thì sẽ được xếp lên trên và được ưu tiên mua vé trước. Quay trở lại với ngữ cảnh của giải thuật này, một bộ xử lý có thể được phân công xử lý nhiều công việc cùng lúc vì vậy nó cần có một lịch trình cụ thể để thực thi các công việc trên. Nếu sử dụng chiến lược FCFS để lập lịch trình thì công việc nào được gửi đến bộ xử lý trước thì sẽ được xử lý trước và những công việc đến sau sẽ phải đợi cho đến khi công việc hiện tại hoàn thành thì mới được xử lý.

*2.2 Mô tả thuật toán*

Thuật toán FCFS có thể được mô tả như sau:

1. Tất cả các tiến trình được xếp hàng đợi theo thứ tự chúng đến.
2. Khi CPU trống, tiến trình ở đầu hàng đợi sẽ được phân bổ cho nó.
3. Tiến trình đang chạy sẽ tiếp tục chạy cho đến khi nó hoàn tất hoặc bị chặn.
4. Khi một tiến trình đang chạy bị chặn, nó sẽ được đưa trở lại cuối hàng đợi.

*2.3 Nguyên tắc hoạt động*

Lập lịch First-Come, First-Served (FCFS) là một trong những thuật toán lập lịch đơn giản nhất trong hệ thống điều hành. Cơ sở của nó dựa trên nguyên tắc “đến trước, phục vụ trước”. Điều này có nghĩa là các tiến trình sẽ được xử lý theo thứ tự chúng đến và yêu cầu CPU. Không có sự phân biệt hay ưu tiên nào giữa các tiến trình; tiến trình đến đầu tiên sẽ được phục vụ đầu tiên.

*2.4 Ưu điểm và nhược điểm*

- Ưu điểm: Thuật toán FCFS, với cơ chế “người đến trước, người phục vụ đầu tiên”, mang lại nhiều ưu điểm trong việc lập lịch hệ thống. Đầu tiên và quan trọng nhất, đó là sự đơn giản của nó. FCFS không yêu cầu tính toán phức tạp hay quản lý dữ liệu phức tạp, làm cho việc triển khai và bảo trì trở nên dễ dàng hơn. Sự đơn giản này cũng làm tăng tính minh bạch trong quá trình lập lịch, nơi mỗi tiến trình được xử lý một cách công bằng dựa trên thời gian đến của nó, không ưu tiên tiến trình nào.

- Nhược điểm: Tuy nhiên, bên cạnh những ưu điểm, FCFS cũng có những hạn chế đáng kể. Một trong những nhược điểm chính là thiếu tính linh hoạt và không có khả năng ưu tiên. Trong FCFS, mỗi quá trình sẽ chạy đến khi hoàn thành, mà không xem xét đến độ ưu tiên hay cấp độ quan trọng của nó. Điều này có thể dẫn đến tình trạng chết đói (starvation), nơi các tiến trình ngắn nhưng có độ ưu tiên cao bị trì hoãn không cần thiết bởi các tiến trình dài hơn đang chiếm dụng CPU.

*2.5 So sánh giải thuật FCFS với các giải thuật lập trình CPU khác (như SJF)*

- FCFS: Lựa chọn quá trình theo thứ tự đến trước. Thời gian chờ có thể cao và không đảm bảo hiệu suất tối ưu. Công bằng trong việc xử lý quá trình theo thứ tự đến.

- SJF: Lựa chọn quá trình có thời gian thực thi ngắn nhất. Giảm thiểu thời gian chờ và cải thiện hiệu suất tổng thể. Không đảm bảo công bằng tuyệt đối.

- RR: Chia thời gian xử lý thành các time slice nhất định và lựa chọn quá trình tuần tự để thực thi trong mỗi time slice. Giảm thiểu thời gian chờ bằng cách phân chia công bằng thời gian xử lý. Đảm bảo công bằng trong việc xử lý các quá trình.

🡪 Tóm lại, FCFS đơn giản nhưng có thời gian chờ cao và không đảm bảo hiệu suất tối ưu. SJF giảm thiểu thời gian chờ và cải thiện hiệu suất, nhưng không đảm bảo công bằng tuyệt đối. RR giảm thiểu thời gian chờ và đảm bảo công bằng bằng cách phân chia thời gian xử lý công bằng cho mỗi quá trình. Lựa chọn giải thuật phụ thuộc vào yêu cầu và tính chất của hệ thống, như thời gian chờ, hiệu suất và công bằng.

*2.6 Kết luận*

\* Tối ưu hiệu suất hệ thống:

Giảm thời gian chờ trung bình cho các yêu cầu, nâng cao trải nghiệm người dùng và hiệu quả hoạt động của hệ thống.

Tối ưu hóa việc sử dụng tài nguyên hệ thống, đặc biệt là bộ nhớ và CPU, tiết kiệm chi phí vận hành.

Đảm bảo đáp ứng các yêu cầu thời gian thực một cách hiệu quả, đặc biệt quan trọng trong các ứng dụng nhúng và hệ thống điều khiển.

\* Nâng cao khả năng mở rộng hệ thống:

Cho phép xử lý lượng lớn yêu cầu một cách hiệu quả, đáp ứng nhu cầu tăng trưởng của hệ thống trong tương lai.

Giảm thiểu tắc nghẽn hệ thống, giúp hệ thống hoạt động ổn định và trơn tru dưới tải trọng cao.

Tăng khả năng thích ứng với các thay đổi trong môi trường hệ thống, đảm bảo hệ thống hoạt động hiệu quả trong nhiều trường hợp khác nhau.

\* Tăng cường tính linh hoạt và khả năng kiểm soát:

Cung cấp cho lập trình viên khả năng tùy chỉnh giải thuật FCFS để đáp ứng các nhu cầu cụ thể của ứng dụng.

Cho phép tích hợp giải thuật FCFS với các giải thuật lập lịch khác để tối ưu hóa hiệu suất cho các trường hợp sử dụng khác nhau.

Tăng cường khả năng kiểm soát hành vi của hệ thống, giúp lập trình viên dễ dàng theo dõi và khắc phục sự cố.

\* Phát triển kỹ năng lập trình C++:

Việc lập trình giải thuật FCFS hiệu quả bằng C++ giúp rèn luyện các kỹ năng lập trình C++ quan trọng như quản lý bộ nhớ, lập trình hướng đối tượng, tối ưu hóa hiệu suất và lập trình hệ thống.

Nâng cao khả năng tư duy logic, giải quyết vấn đề và thiết kế hệ thống hiệu quả.

Cung cấp cho lập trình viên nền tảng vững chắc để phát triển các ứng dụng hệ thống phức tạp và hiệu quả.

\* Ứng dụng rộng rãi:

Giải thuật FCFS được sử dụng trong nhiều lĩnh vực khác nhau như hệ điều hành (Linux, Windows, macOS), mạng máy tính (TCP/IP), hệ thống nhúng (thiết bị IoT, robot), và mô phỏng hệ thống.

Việc thành thạo lập trình giải thuật FCFS bằng C++ giúp lập trình viên có nhiều cơ hội việc làm trong các lĩnh vực này.

🡪 Kết luận:

Lập trình giải thuật FCFS hiệu quả bằng C++ là một kỹ năng quan trọng cho bất kỳ lập trình viên hệ thống nào. Việc nắm vững kỹ năng này giúp nâng cao hiệu suất hệ thống, tăng cường khả năng mở rộng, linh hoạt và kiểm soát, đồng thời phát triển kỹ năng lập trình C++ và mở ra nhiều cơ hội việc làm trong tương lai.

**Chương 3. Cài đặt giải thuật FCFS bằng C++**

*3.1 Các bước cài đặt thuật toán FCFS*

Giải thích từng bước mã nguồn C++ để triển khai giải thuật FCFS.

Mô tả các biến, hàm và cấu trúc dữ liệu được sử dụng trong chương trình.

Giải thích cách thức chương trình xử lý các yêu cầu CPU theo thuật toán FCFS.

Chương trình trên triển khai giải thuật FCFS (First-Come, First-Served) để lập lịch CPU. Dưới đây là giải thích từng bước của mã nguồn C++:

B1: Đầu tiên, chương trình bắt đầu bằng việc khai báo và định nghĩa một lớp `Process` để biểu diễn thông tin về các tiến trình. Lớp này chứa các thành phần dữ liệu như `arrivalTime` (thời gian đến), `burstTime` (thời gian thực thi), `waitingTime` (thời gian chờ đợi), `turnaroundTime` (thời gian quay vòng), `cpTime` (thời gian tiến trình kết thúc thực thi) và `ten` (số hiệu của tiến trình).

B2: Sau đó, chương trình định nghĩa một số hàm hỗ trợ. Hàm `calculateWating\_CPTime` tính thời gian chờ đợi và thời gian kết thúc thực thi của mỗi tiến trình. Hàm `calculateTurnarounddTime` tính thời gian quay vòng của mỗi tiến trình. Hàm `calculateAverageTime` tính trung bình thời gian chờ đợi và thời gian quay vòng của tất cả các tiến trình. Hàm `cmp` là một hàm so sánh được sử dụng để sắp xếp các tiến trình theo thứ tự đến trước phục vụ trước.

B3: Hàm `fcfs` chính là hàm thực hiện giải thuật FCFS. Trong hàm này, người dùng được yêu cầu nhập số lượng tiến trình và thông tin về thời gian đến (`arrivalTime`) và thời gian thực thi (`burstTime`) của từng tiến trình. Sau đó, danh sách các tiến trình được sắp xếp theo thứ tự đến trước phục vụ trước bằng cách gọi hàm `sort` với hàm so sánh `cmp`.

B4: Tiếp theo, chương trình tính toán thời gian chờ đợi và thời gian kết thúc thực thi của từng tiến trình thông qua hàm `calculateWating\_CPTime`.

B5: Sau khi tính toán xong, chương trình tính thời gian quay vòng của mỗi tiến trình bằng cách gọi hàm `calculateTurnarounddTime`.

B6: Tiếp theo, chương trình tính trung bình thời gian chờ đợi và thời gian quay vòng của tất cả các tiến trình bằng cách gọi hàm `calculateAverageTime`. Kết quả được in ra màn hình.

B7: Cuối cùng, chương trình in biểu đồ Gantt để hiển thị thời gian tiến trình kết thúc thực thi của từng tiến trình. Biểu đồ Gantt được in ra màn hình dưới dạng các thanh ngang, mỗi thanh đại diện cho một tiến trình.

\* Tóm lại, chương trình trên triển khai giải thuật FCFS bằng cách tính toán thời gian chờ đợi, thời gian quay vòng và thời gian kết thúc thực thi của từng tiến trình. Kết quả được in ra màn hình, bao gồm trung bình thời gian chờ đợi và thời gian quay vòng cũng như biểu đồ Gantt để minh họa thời gian tiến trình kết thúc thực thi.

*3.2 Code*

*Vì code khá dài nên em xin phép được đẩy code lên github.  
https://github.com/thien349k/Algorithm-FCFS*

*3.3 Kiểm thử*

