

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

-----o0o-----



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: NGUYÊN LÝ HỆ ĐIỀU HÀNH**

**ĐỀ TÀI: LẬP TRÌNH MÔ PHỎNG CÁC PHƯƠNG PHÁP LẬP LỊCH CHO CPU**

GVHD: Nguyễn Bá Nghiễn

Nhóm thực hiện: Nhóm 9

Thành viên nhóm:

Bùi Huy Hải: 2021600856

Trần Đình Hoan: 2021602715

Đặng Văn Hoàng: 2021606501

Hoàng Tuấn Linh: 2021607036

Trần Duy Toàn: 2021601204

*Hà Nội, 2023*

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 2](#_Toc136307145)

[**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT** 3](#_Toc136307146)

[**1.1.Giới thiệu** 3](#_Toc136307147)

[**1.1.1.** **Mục tiêu lập lịch** 3](#_Toc136307148)

[**1.1.2.** **Các đặc điểm của tiến trình** 3](#_Toc136307149)

[**1.1.3.** **Điều phối không độc quyền và điều phối độc quyền** 5](#_Toc136307150)

[**1.2.Các khái niệm cơ bản** 6](#_Toc136307151)

[**1.2.1. Khái niệm giờ CPU** 6](#_Toc136307152)

[**1.2.2. Các trạng thái của tiến trình liên quan đến giờ CPU** 6](#_Toc136307153)

[**1.2.3. Khái niệm lập trình cho CPU** 8](#_Toc136307154)

[**1.3. Các thuật toán lập lịch** 9](#_Toc136307155)

[**1.3.1. First Come First Served(FCFS)** 9](#_Toc136307156)

[**1.3.2. Round robin(RR)** 10](#_Toc136307157)

[**1.3.3. Shortest Job First(SJF)** 11](#_Toc136307158)

[**1.3.4. Shortest Remain Time(SRT)** 12](#_Toc136307159)

[**CHƯƠNG 2. CÀI ĐẶT THUẬT TOÁN** 13](#_Toc136307160)

[**2.1. Mô hình cài đặt thuật toán** 13](#_Toc136307161)

[**2.1.1. Cấu trúc dữ liệu** 13](#_Toc136307162)

[**2.1.2. Thuật toán xử lý chung** 14](#_Toc136307163)

[**2.2. Thuật toán** 15](#_Toc136307164)

[**2.2.1. Hàm main()** 15](#_Toc136307165)

[**2.2.2. First Come First Served(FCFS)** 17](#_Toc136307166)

[**2.2.3. Round robin(RR)** 17](#_Toc136307167)

[**2.2.4. Shortest Job First(SJF)** 18](#_Toc136307168)

[**2.2.5. Shortest Remain Time(SRT)** 18](#_Toc136307169)

[**CHƯƠNG 3. KẾT LUẬN** 18](#_Toc136307170)

[**3.1. Kết quả** 18](#_Toc136307171)

[**3.2. Tài liệu tham khảo** 18](#_Toc136307172)

# **LỜI NÓI ĐẦU**

Hệ điều hành là phần gắn bó trực tiếp với phần cứng và là môi trường để cho các chương trình ứng dụng khác chạy trên nó. Với chức năng quản lý và phân phối tài nguyên một cahcs hợp lý, đồng thời giả lập một máy tính mở rộng và tạo giao diện tiện lợi với người dùng, hệ điều hành là một thành phần then chốt không thể thiếu được trong mỗi một hết thống máy tính điện tử.

Một trong những chức năng quan trọng của hệ điều hành là quản lý CPU. Trong môi trường xử lý đa chương, có thể xảy ra tình huống nhiều tiến trình đổng thời sẵn sàng để xử lý. Mục tiêu cảu các hệ phân chia thời gian(time-sharing) la chuyển đổi CPU qua lại giữa các tiến trình một cách thường xuyên để nhiều người sử dụng có thể tương tác cùng lúc với từng chương trình trong quá trình xử lý.

Để thực hiện được mục tiêu này, hệ điều hành phải lựa chọn tiến trình được xử lý tiếp theo. Bộ điều phối sẽ sử dụng một giải thuật điều phối thích hợp để thực hiện nhiệm vụ này. Một thành phần khác của hệ điều hành cũng tiềm ẩn trong công tác điều phối là bộ điều phối(dispatcher). Bộ phân phối sẽ chịu trách nhiệm chuyển đổi ngữ cảnh và trao CPU cho tiến trình được chọn bởi bộ điều phối để xử lý.

Vì những lợi ích lớn lao mà giải thuật điều phối CPU đem lại và để tìm hiểu kĩ hơn về nguyên tắc hoạt động của chúng, chúng em sẽ cùng nhau tìm hiểu, nghiên cứu đề tài: Lập trình mô phỏng các phương pháp lập lịch cho CPU.

Nhóm 9 sẽ trình bày nội dung chính như sau:

* Tìm hiểu các thuật toán: First Come First Served(FCFS), Round Robin(RR), Shortest Job First(SJF), Shortest Remain Time(SRT).
* Chỉ ra được ưu và nhược điểm của các thuật toán lập lịch CPU
* Minh họa qua một bài toán cụ thể.

Mặc dù nhóm 9 chúng em đã cố gắng tìm hiểu kiến thức, tài liệu về các phương pháp lập lịch CPU, cũng như cố gắng trong quá trình hoàn thiện bài tập. Song do kiến thức, trình độ còn nhiều hạn chế nên chắc chắn không tránh khỏi những sai sót. Chúng em mong sự chỉ dẫn, góp ý của thầy giáo để chúng em có thể hiểu và hoàn thiện hơn bài tập lớn này.

# **CHƯƠNG 1. CƠ SỞ LÝ THUYẾT**

## **1.1.Giới thiệu**

**Yêu cầu đề tài:** Trình bày các thuật toán lập lịch CPU đồng thời lập trình mô phỏng, minh họa qua một bài toán lập lịch CPU cụ thể.

### **Mục tiêu lập lịch**

Bộ điều phối không cung cấp cơ chế mà đưa ra các quyết định. Các hệ điều hành xây dựng nhiều chiến lược khác nhau để thực hiện việc điều phối, nhưng tựu chung cần đạt được các mục tiêu sau:

* Sự công bằng: Các tiến trình chia sẽ CPU một cách công bằng không có tiến trình nào phải đợi vô hạn để được cấp phát CPU.
* Tính hiệu quả: Hệ thống phải tận dụng được CPU 100% thời gian.
* Thời gian đáp ứng hợp lý: Cực tiểu hóa thời gian hoàn tất các tác vụ xử lý theo lô.
* Thông lượng tối đa: Cực đại hóa số công việc được xử lý trong một đơn vị thời gian.

Tuy nhiên thường không thể thỏa mãn các mục tiêu kể trên vì bản thân chúng có sự mâu thuẫn với nhau mà chỉ có thể dung hòa chúng ở một mức độ nào đó.

### **Các đặc điểm của tiến trình**

Điều phối hoạt động của các tiến trình là một vấn đề rất phức tjap, đòi hỏi hệ điều hành khi giải quyết phải xem xét nhiều yếu tố khác nhau để có thể đạt được những mục tiêu đề ra. Một số đặc tính của tiến trình cần được quan tâm như tiêu chuẩn điều phối:

* Tính hướng xuất/nhập của tiến trình: Khi một tiến trình được nhận CPU, chủ yếu nó chỉ sử dụng CPU đến khi phát sinh một yêu cầu nhập xuất. Hoạt động của các tiến trình như thế thường bao gồm nhiều lượt sử dụng CPU, mỗi lượt trong một thời gian khá ngắn.
* Tính hướng xử lý của tiến trình: Khi một tiến trình được nhận CPU, nó có khuynh hướng sử dụng CPU đến khi hết thời gian dành cho nó? Hoạt động của các tiến trình như thế thường bao gồm một số ít lượt sử dụng CPU, nhưng mỗi lượt trong một thời gian đủ dài.
* Tiến trình tương tác hay xử lý theo lô: Người sử dụng theo kiểu tương tác thường yêu cầu được hồi đáp tức thời đối với các yêu cầu của họ, trong khi các tiến trình của các tác vụ được xử lý theo lô nói chung có thể trì hoãn trong một thời gian chấp nhận được.
* Độ ưu tiên của tiến trình: Các tiến trình có thể được phân cấp theo một số tiêu chuẩn dánh giá nào đó, một cách hợp lý, các tiến trình quan trọng hơn(có độ ưu tiên cao hơn) cần được ưu tiên cao hơn.
* Thời gian đã sử dụng CPU của tiến trình: Một số quan điểm ưu tiên chọn những tiến trình đã sử dụng CPU nhiều thời gian nhất vì hy vọng chúng sẽ cần ít thời gian nhất để hoàn tất và rời khỏi hệ thống. Tuy nhiên cũng có qua điểm cho rằng các tiến trình nhận được CPU trong ít thời gian là những tiến trình đã phải chờ lâu nhất, do vậy ưu tiên chọn chúng.
* Thời gian còn lại tiến trình cần để hoàn tất: Có thể giam thiểu thời gian chờ trung bình của các tiến trình bằng cách cho các tiến trình cần ít thời gian nhất để hoàn tất được thực hiện trước. Tuy nhiên đáng tiếc là rất hiếm khi biết được tiến trình cần bao nhiêu thời gian nữa để kết thúc xử lý.

### **Điều phối không độc quyền và điều phối độc quyền**

Thuật toán điều phối cần xem xét và quyết định thời điểm chuyển đổi CPU giữa các tiến trình. Hệ điều hành có thể thực hiện cơ chế điều phối theo nguyên lý độc quyền hoặc không độc quyền:

Điều phối độc quyền: Nguyên lý điều phối độc quyền cho phép một tiến trình khi nhận được CPU sẽ có quyền độc chiếm CPU đến khi hoàn tất xử lí hoặc tự nguyện giải phóng CPU. Khi đó quyết định điều phối CPU sẽ xảy ra trong các tình huống sau:

* Khi tiến trình chuyển từ trạng thái đang xử lý (running) sang trạng thái bị blocked (ví dụ chờ một thao tác nhập xuất hay chờ một tiến trình con kết thúc…)
* Khi tiến trình kết thúc.

Điều phối không độc quyền: Ngược với nguyên lý độc quyền, điều phối theo nguyên lý không độc quyền cho phép tạm dừng hoạt động của một tiến trình sẵn sàng xử lý. Khi một tiến trình nhận được CPU, nó vẫn được sử dụng CPU đến khi hoàn tất hoặc tự nguyện giải phóng CPU, nhưng khi một tiến trình khác có độ ưu tiên có thể dành quyền sử dụng CPU của tiến trình ban đầu. Như vậy là tiến trình có thể bị tạm dừng hoạt động bất cứ lúc nào mà không được báo trước, để tiến trình khác xử lý. Các quyết địng điều phối xảy ra khi:

* Khi tiến trình chuyển từ trạng thái đang xử lý (running) sang trạng thái bị blocked.
* Khi tiến trình chuyển từ trạng thái đang xử lý (running) sang trạng thái ready (ví dụ: một thao tác nhập xuất hoàn tất).
* Khi tiến trình kết thúc.

Trong các hệ thống sử dụng nguên lý điều phối độc quyền có thể xảy ra tình trạng các tác vụ cần thời gian xử lý ngắn phải chờ tác vụ xử lý với thời gian rất dài hoàn tất. Nguyên lý điều phối độc quyền thường chỉ thích hợp với các hệ xử lý theo lô.

Đối với các hệ thống tương tác (time sharing), các hệ thời gian thực (real time), cần phải sử dụng nguyên lý điều phối không độc quyền để các tiến trình quan trọng có cơ hội hồi đáp kịp thời. Tuy nhiên thực hiện điều phối theo nguyên lý không độc quyền đòi hỏi những cơ chế phức tạp trong việc phân định rõ độ ưu tiên, và phát sinh thêm chi phí khi chuyển đổi CPU qua lại giữa các tiến trình.

## **1.2.Các khái niệm cơ bản**

### **1.2.1. Khái niệm giờ CPU**

CPU là một loại tài nguyên quan trọng của máy tính. Mọi tiến trình muốn hoạt động được đều phải có sự phục vụ của CPU(để xử lý, tính toán…). Thời gian mà CPU phục vụ cho tiến trình hoạt động gọi là giờ CPU.

Tại mỗi thời điểm, chỉ có một tiến trình được phân phối giờ CPU để hoạt động(thực hiện các lệnh của mình).

### **1.2.2. Các trạng thái của tiến trình liên quan đến giờ CPU**

Trong chế độ đa chương trình, có ba trạng thái của tiến trình liên quan mật thiết đến giờ CPU bao gồm:

A picture containing circle, diagram, line, white

Description automatically generated

***Hình 1.1: Các trạng thái của tiến trình liên quan đến giờ CPU***

Sẵn sàng(ready): Là trạng thái mà tiến trình được phân phối đầy đủ mọi tài nguyên cần thiết và đang chờ giờ CPU.

Thực hiện(running): Là trạng thái mà tiến trình được phân phối đầy đủ mọi tài nguyên cần thiết và giờ CPU.

Đợi(waiting): Là trạng thái tiến trình không thực hiện được vì thiếu một vài điều kiện nào đó(đợi dữ liệu vào/ra, đợi tài nguyên bổ sung…). Khi sự kiện mà nó chờ đợi xuất hiện, tiến trình sẽ quay lại trạng thái sẵn sàng.

Như vậy, trong suốt thời gian tồn tại của mình, các tiến trình sẽ tuân thủ theo sơ đồ thực hiện sau:

A picture containing text, receipt, font, line

Description automatically generated

***Hình 1.2: Sơ đồ thực hiện các tiến trình***

Một tiến trình đang trọng trạng thái thực hiện, nó có thể rời khỏi trạng thái bởi một trong ba lý do:

* Tiến trình đã hoàn thành công việc, khi đó nó trả lại giờ CPU và chuyển sang chờ xử lý kết thúc.
* Tiến trình tự ngắt: 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 kiến nó đang chờ.
* Tiến trình sử dụng hết giờ CPU dành cho nó, khi đó nó sẽ được chuyển sang trạng thái sẵn sàng.

Việc chuyển tiến trình sang trạng thái sẵn sàng về bản chất là thực hiện việc phân phối lại giờ CPU.

### **1.2.3. Khái niệm lập trình cho CPU**

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. Các từ trạng thái được tổ chức theo kiểu hàng đợi như sau:

A picture containing diagram, text, plan, line

Description automatically generated

***Hình 1.3: Sơ đồ tổ chức hàng đợi các tiến trình***

Như vậy lập lịch cho CPU có nghĩa là tổ chức một hàng đợi các tiến trình sẵn sàng để phân phối giờ CPU cho chúng dựa trên độ ưu tiên của các tiến trình, sao cho hiệu suất sử dụng CPU 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.

## **1.3. Các thuật toán lập lịch**

### **1.3.1. 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. Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt.

A picture containing screenshot, graphics, text, diagram

Description automatically generated

***Hình 1.4: Điều phối FIFO***

* Ưu điểm:
* Giờ CPU không bị phân phối lại(không bị ngắt) và chỉ phải 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:
* Thời gian chờ trung bình sẽ tăng vô hạn khi hệ thống tiếp cận tới hạn khả năng phục vụ của mình.
* Nếu độ phát tán thời gian thực hiện tiến trình tăng thì thời gian chờ đợi trung bình cũng tăng theo.
* Khi có tiến trình dài, ít bị ngắt thì các tiến trình khác phải chờ đợi lâu hơn.

### **1.3.2. 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 FIFO nhưng sự trưng dụng CPU được thêm vào để chuyển CPU giữa các quá trình. Đơn vị thời gian nhỏ được gọi là định mức thời gian(time quantum) hay phần thời gian (timeslice) được định nghĩa. Định mức thời gian thường từ 10 đến 100 mili giây. Hàng đợi sẵn sàng được xem như một hàng đợi vòng. Bộ định thời CPU di chuyển vòng quanh hàng đợi sẵn sàng, cấp phát CPU tới mỗi quá trình có khoảng thời gian tối đa bằng một định mức thời gian.

Để cài đặt định thời RR, chúng ta quản lý hàng đợi sẵn sàng như một hàng đợi

FIFO của các quá trình. Các quá trình mới được thêm vào đuôi hàng đợi. Bộ định

thời CPU chọn quá trình đầu tiên từ hàng đợi sẵn sàng, đặt bộ đếm thời gian để ngắt

sau 1 định mức thời gian và gửi tới quá trình.

Sau đó, một trong hai trường hợp sẽ xây ra. Quá trình có 1 chu kỳ CPU ít hơn 1

định mức thời gian. Trong trường hợp này, quá trình sẽ tự giải phóng . Sau đó, bộ

định thời biểu sẽ xử lý quá trình tiếp theo trong hàng đợi sẵn sàng. Ngược lại, nếu

chu kỳ CPU của quá trình đang chạy dài hơn 1 định mức thời gian thì bộ đếm thời

gian sẽ báo và gây ra một ngắt tới hệ điều hành. Chuyên đổi ngữ cảnh sẽ được thực

thì và quá trình được đặt trở lại tại đuôi của hàng đợi sẵn sàng. Sau đó, bộ định thời

biểu CPU sẽ chọn quá trình tiếp theo trong hàng đợi sẵn sàng.

A picture containing screenshot, diagram, line, rectangle

Description automatically generated

***Hình 1.5: Round Robin***

* Ưu điểm:
* Các quá trình sẽ được luân phiên cho CPU xử lý nên thời gian chờ đợi sẽ ít.
* Đối với các quá trình liên quan đến nhập xuât(I/O), người dùng thì rất hiệu quả.
* Việc cài đặt không quá phức tạp
* Nhược điểm:
* Thời gian chờ đợi trung bình dưới chính sách RR thường là quá dài.
* Nếu thời gian định mức cho việc xử lý quá lớn thì RR thành FIFO.
* Nếu thời gian 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 định mức thời gian nên dài hơn 80% chu kì CPU.

### **1.3.3. Shortest Job First(SJF)**

Một tếp cận khác đối với việc định thời CPU là giải thuật định thời công việc

ngắn nhất trước (shortest-job-first-SJF). Giải thuật này gán tới mỗi quá trình chiều

dài của chu kỳ CPU tiếp theo cho quá trình sau đó. Khi CPU sẵn dùng, nó được gán

tới quá trình có chu kỳ CPU kế tiếp ngắn nhất. Nếu hai quá trình có cùng chiều dài

chu kỳ CPU kế tiếp, định thời FIFO được dùng. Chú ý rằng thuật ngữ phù hợp hơn

là chu kỳ CPU kế tiếp ngắn nhất (shortest next CPU burst) vì định thời được thực

hiện bằng cách xem xét chiều dài của chu kỳ CPU kế tiếp của quá trình hơn là toàn

bộ chiều dài của nó. Chúng ta dùng thuật ngữ SJF vì hầu hết mọi người và mọi sách

tham khảo tới nguyên lý của loại định thời biêu này như SJF.

* Ưu điểm:
* Giải thuật được xem là tối ưu, thời gian chờ đợi trung bình giảm
* Tận dụng hết năng lực của CPU
* Nhược điểm :
* Cài đặt thuật toán phức tạp,tồn nhiều xữ lý cho quá trình quản lý.
* Mặc dù SJF là tối ưu nhưng nó không thê được cài đặt tại cấp định thời CPU ngắn vì không có cách nào đề biết chiều dài chu kỳ CPU tiếp theo.
* Giải thuật SJF có thể trưng dụng hoặc không trưng dụng CPU, dẫn tới giảithuật này có nhiều dị bản khác nhau và sẽ tối ưu hay không tối ưu phụ thuộc vào trưng dụng CPU.

### **1.3.4. 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 nốt 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 giờ CPU cũng phải được áp dụng nếu không sẽ làm mất tính ưu việt của thuật toán.

* Ưu điểm:
* Thời gian chờ đợi, tồn tại trong hệ thống của mỗi tiến trình đều ngắn.
* Thuật toán tối ưu nhất.
* 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 đến của mỗi tiến trình.

# **CHƯƠNG 2. CÀI ĐẶT THUẬT TOÁN**

## **2.1. Mô hình cài đặt thuật toán**

### **2.1.1. Cấu trúc dữ liệu**

Cấu trúc dữ liệu đề xuất cho tiến trình được xây dựng thành các struct nhằm tạo điều kiện thuật lợi cho việc quản lý các tiến trình.

1. ***Cấu trúc dữ liệu tiến trình***

Để lưu trữ thông tin về tiến trình dùng một struct “process” gồm có 4 trường:

A picture containing text, font, screenshot, white

Description automatically generated

* id: tên tiến trình
* arrivalTime: thời gian đến
* burstTime: thời gian xử lý của tiến trình
* waitingTime: thời gian chờ

1. ***Đầu vào:***

Một danh sách gồm các tiến trình.

1. ***Đầu ra:***

Một danh sách bao gồm các struct “Schedule” dùng để vẽ sơ đồ giant mỗi struct Schedule là một ô được vẽ trong sơ đồ Giant

Struct “Schedule” gồm các trường như sau:

A picture containing text, screenshot, font, white

Description automatically generated

* processId[]: tên các tiến trình
* startTime[]: Thời gian bắt đầu
* totalSlots: số lượng các ô được vẽ

Ví dụ để có được sơ đồ Giant như sau:

A picture containing screenshot, line, text, font

Description automatically generated

Thì đầu ra cần có G. processId[1]=1,G. processId[2]=2,G. processId[3]=3

G. startTime[1]=0,G. startTime[2]=1,G. startTime[3]=10,G. totalSlots=3.

### **2.1.2. Thuật toán xử lý chung**

Việc cài đặt thuật toán được mô phỏng theo cách làm việc của CPU và tất cả thuật toán con đều theo mô hình thuật toán này.

* Tiến trình ở đầu danh sách sẽ được ưu tiên xử lý trước và nó chiếm dụng CPU tại thời điểm đó.
* Việc đi kèm theo là xem xét thời gian xử lý các tiến trình đã hết chưa. Nếu đã hết thì nghĩa là hoàn thành việc xử lý, ngược lại thì tiếp tục xử lý theo thuật toán.
* Xong mỗi chu kỳ của CPU (1 quantum) thì cập nhật lại danh sách để loại bỏ các tiến trình đã hoàn thành hay sắp xếp thêm các tiến trình mới vào.

A picture containing text, screenshot, diagram, font

Description automatically generated

***Hình 2.1: Sơ đồ thuật toán đề xuất chung cho các giải thuật***

## **2.2. Thuật toán**

Trước khi thực hiện các thuật toán thì các tiến trình được sắp xếp theo thời gian đến tăng dần.

### **2.2.1. Hàm main()**

**Code:**

**A screen shot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer code

Description automatically generated with low confidence**

### **2.2.2. First Come First Served(FCFS)**

Đối với thuật toán này thứ tự thực hiện ưu tiên theo thời gian xuất hiện.Tiến trình nào đến trước sẽ thực hiện trước.

***Hình 2.2: Sơ đồ khối FCFS***

**Code:**

### **2.2.3. Round robin(RR)**

Đối với thuật toán này quá trình thực hiện sẽ phân theo lượng tử (quantumn) thời gian q .Các tiến trình được đưa vào một hàng đợi và chờ được thực hiện. Theo thời gian xuất hiện, thực hiện của tiến trình với một khoảng thời gian q, nếu tiến trình đó không thực hiện xong với khoản thời gian q đó sẽ bị đưa xuống cuối hàng đợi. Trong khi nó đang thực hiện, nếu có tiến trình nào xuất hiện nó sẽ được đưa vào đầu hàng đợi sau khi thực hiện xong quantumn.

A picture containing text, diagram, plan, post-it note

Description automatically generated

***Hình 2.3: Sơ đồ khối RR***

**Code:**

### **2.2.4. Shortest Job First(SJF)**

Đối với thuật toán này thứ tự thực hiện ưu tiên theo thời gian thực hiện công việc. Tiến trình nào có thời gian làm việc với CPU nhỏ nhất sẽ ưu tiên thực hiện trước. Trong trường hợp thời gian làm việc của các tiến trình bằng nhau thì sẽ xử lý theo thuật toán FCFS.

***Hình 2.4: Sơ đồ khối SJF***

**Code:**

### **2.2.5. Shortest Remain Time(SRT)**

Trong thuật toán này thứ tự thực hiện sẽ ưu tiên theo theo thời gian thực hiện công việc còn lại. Tiến trình nào có thời gian thực hiện còn lại nhỏ nhất sẽ thực hiện trước.

***Hình 2.3: Sơ đồ khối SRT***

**Code:**

# **CHƯƠNG 3. KẾT LUẬN**

### **3.1. Kết quả**

Trong thời gian thực hiện đề tài em đã hiểu hơn về cơ chế và cách thức làm việc của hệ điều hành đồng thời có thêm về kinh nghiệm lập trình cấu trúc trong C++ . Trong quá trình thực hiện em xin chân thành cảm ơn sự chỉ dạy của thầy ***Nguyễn Bá Nghiễn***. Dù đã cố gắng hết mình nhưng bài làm không thể tránh khỏi thiếu sót, em mong nhận được sự cảm thông và góp ý của quý thầy cô và các bạn.

### **3.2. Tài liệu tham khảo**

[1] Vũ Lê Hùng, Giáo Trình Nguyên Lý Hệ Điều Hành, Đại học Bách Khoa

Hồ Chí Minh.

[2] Lê Trung Dũng, Giáo Trình Hệ Điều Hành, Nhà xuất bản Giáo Dục.

[3] Trần Hồ Thủy Tiên.Giáo Tìình Nguyên Lý Hệ Điều Hành.Đại học Đà Nẵng,

Trường đại học Bách Khoa, Khoa Công Nghệ Thông Tin 01/04/2010.

[4] [Link tai lieu tham khao](Link%20tai%20lieu%20tham%20khao)