BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP CƠ SỞ**

**CÀI ĐẶT VÀ MÔ PHỎNG CÁC BƯỚC THỰC HIỆN TÍNH TOÁN THỜI GIAN CHỜ CỦA CÁC TIẾN TRÌNH TRONG GIẢI THUẬT FCFS (First Come First Serve) VÀ SJF (Shortest Job First) SỬ DỤNG GIAO DIỆN TEXT VÀ ĐỒ HỌA**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | Nguyễn Khắc Cường |
| Sinh viên thực hiện: | Nguyễn Sanh Quốc Huy |
| Mã số sinh viên: | 62130757 |

Khánh Hòa – Tháng 12/2022

# LỜI CẢM ƠN

Để hoàn thành đề tài thực tập cơ sở này, trước hết em xin gửi đến quý thầy, cô Khoa Công nghệ Thông tin - Trường Đại học Nha Trang lời cảm ơn chân thành.

Em xin gửi đến thầy Nguyễn Khắc Cường, là người hướng dẫn, giúp đỡ em hoàn thành đề tài thực tập cơ sở này lời cảm ơn sâu sắc nhất.

Trong quá trình nghiên cứu thực hiện đề tài, cũng như là trong quá trình làm bài báo cáo tốt nghiệp, khó tránh khỏi sai sót, rất mong các thầy, cô bỏ qua. Đồng thời do kiến thức cũng như kinh nghiệm thực tiễn của bản thân còn hạn chế nên đề tài, bài báo cáo thực tập này khó thể không tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp từ quý thầy, cô để em học hỏi thêm được nhiều kinh nghiệm, cũng như kỹ năng cần thiết.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[LỜI CẢM ƠN 2](#_Toc123643577)

[MỞ ĐẦU 4](#_Toc123643578)

[Chương 1. TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU 7](#_Toc123643579)

[1.1 CƠ SỞ LÝ THUYẾT 7](#_Toc123643580)

[1.1.1 Giới thiệu 7](#_Toc123643581)

[1.1.1.1 Khái niệm hệ điều hành 7](#_Toc123643582)

[1.1.1.2 Các chức năng chính của hệ điều hành 7](#_Toc123643583)

[1.1.2 Các khái niệm cơ bản 8](#_Toc123643584)

[1.1.2.1 Khái niệm tiến trình và trạng thái của tiến trình 8](#_Toc123643585)

[1.1.2.2 Các đặc điểm của tiến trình 9](#_Toc123643586)

[1.1.2.3 Khái niệm giờ CPU 10](#_Toc123643587)

[1.1.2.4 Các trạng thái của tiến trình liên quan đến giờ CPU 10](#_Toc123643588)

[1.1.2.5 Chu kì sử dụng CPU I/O 11](#_Toc123643589)

[1.1.2.6 Các thông tin cơ bản mà HĐH sử dụng để lập lịch tiến trình 12](#_Toc123643590)

[1.1.2.7 Khái niệm lập lịch cho CPU và lý do cần phải lập lịch CPU 14](#_Toc123643591)

[1.1.2.8 Mục tiêu lập lịch 15](#_Toc123643592)

[1.1.2.9 Điều phối không độc quyền (Nonpreemptive scheduling) và điều phối độc quyền (Preemptive scheduling) 15](#_Toc123643593)

[1.1.2.10 Các bộ điều phối 17](#_Toc123643594)

[1.1.2.11 Khái niệm Dispatcher (bộ điều phối) 18](#_Toc123643595)

[1.1.3 Các giải thuật điều phối CPU cơ bản 19](#_Toc123643596)

[1.1.3.1 First Come First Served (FCFS) 19](#_Toc123643597)

[1.1.3.2 Shortest Job First (SJF) 22](#_Toc123643598)

[1.2 CÁC NGÔN NGỮ ĐƯỢC SỬ DỤNG 24](#_Toc123643599)

[1.2.1 Ngôn ngữ C# (C-Sharp) 24](#_Toc123643600)

[1.2.1.1 Khái niệm 24](#_Toc123643601)

[1.2.1.2 Nền tảng .NET Framework 24](#_Toc123643602)

[1.2.1.3 Ứng dụng của C Sharp 25](#_Toc123643603)

[1.2.1.4 Class trong lập trình hướng đối tượng và C# 26](#_Toc123643604)

[1.2.1.5 Ưu điểm của ngôn ngữ C Sharp là gì? 27](#_Toc123643605)

[Chương 2. CÀI ĐẶT THUẬT TOÁN 29](#_Toc123643606)

[2.1 Cấu trúc chương trình 29](#_Toc123643607)

[2.1.1 Class Tiến trình 29](#_Toc123643608)

[2.1.2 Class danh sách tiến trình 31](#_Toc123643609)

[2.1.3 Class Program 41](#_Toc123643610)

[2.2 Đầu vào và đầu ra của chương trình 43](#_Toc123643611)

[2.3 Các chức năng của chương trình 43](#_Toc123643612)

[2.4 Cài đặt chương trình 43](#_Toc123643613)

[Chương 3. XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG BẰNG GIAO DIỆN 46](#_Toc123643614)

[3.1 Môi trường phát triển 46](#_Toc123643615)

[3.2 Giao diện của chương trình 46](#_Toc123643616)

[3.2.1 Form menu chính 46](#_Toc123643617)

[3.2.2 Form FCFS 47](#_Toc123643618)

[3.2.2.1 FCFS Non Arrival Time 47](#_Toc123643619)

[3.2.2.2 FCFS Arrival Time 48](#_Toc123643620)

[3.2.3 Form SJF 49](#_Toc123643621)

[3.2.3.1 SJF Non Arrival Time 49](#_Toc123643622)

[3.2.3.2 SJF NonPreemptive (Arrival Time) 50](#_Toc123643623)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 51](#_Toc123643624)

[4.1 KẾT QUẢ ĐẠT ĐƯỢC 51](#_Toc123643625)

[4.2 ƯU ĐIỂM 51](#_Toc123643626)

[4.3 HẠN CHẾ 51](#_Toc123643627)

[4.4 HƯỚNG PHÁT TRIỂN 51](#_Toc123643628)

[4.5 KẾT LUẬN 51](#_Toc123643629)

[TÀI LIỆU THAM KHẢO 51](#_Toc123643630)

# MỞ ĐẦU

* **Lý do chọn đề tài:**

Hệ điều hành là thành 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 cách hợp lý, đồng thời giả lập một máy tính mở rộng và tạo giao diện tiện lợi với người sử dụng, hệ điều hành là một thành phần then chốt không thể thiếu được trong mỗi một hệ thống máy tính điện tử. Việc tìm hiểu về hệ điều hành đối với các sinh viên ngành công nghệ thông tin là rất quan trọng. Thông qua môn học “Hệ Điều Hành” em đã được học về các kiến thức cơ bản. Để hiểu rõ hơn về cách thức làm việc, chức năng quản lý và phân phối tài nguyên cũng như ứng dụng các kiến thức đã được học về hệ điều hành vào thực tế em chọn đề tài: “**Cài đặt và mô phỏng các bước thực hiện tính toán thời gian chờ của các tiến trình trong giải thuật** **FCFS (****First Come First Serve) và SJF (Shortest Job First)”.**

* **Mục tiêu nghiên cứu:**
* ***Mục tiêu chung:***
* Tìm hiểu các giải thuật: FCFS (First Come First Serve) và SJF (Shortest Job First).
* Xây dựng chương trình mô phỏng các giải thuật đã tìm hiểu và kết quả demo.
* ***Mục tiêu cụ thể:***
* Chỉ ra được ưu và nhược điềm cả 2 giải thuật lập lịch CPU.
* Các nguyên tắc hoạt động và sự khác nhau của các giải thuật này.
* Xây dựng mô phỏng các bước thực hiện các thuật toán trên bằng ngôn ngữ lập trình (ở đây em chọn ngôn ngữ C# trên môi trường Visual Studio, có sử dụng winform để xây dựng mô phỏng).
* **Nội dung và phương thức thực hiện:**
* ***Nội dung thực hiện:***
* Mức 1:
  + - 1. Trình bày lý thuyết thuật toán FCFS và SJF.
      2. Cài đặt các bước thực hiện thuật toán FCFS và SJF bằng ngôn ngữ lập trình C#.
      3. Dữ liệu thô được nhập từ bàn phím.

Dữ liệu xuất: xuất ra màn hình.

* Mức 2: Mô phỏng bằng đồ họa với windows forms.
* ***Phương pháp:***
* Nghiên cứu giáo trình lập trình hướng đối tượng và hệ điều hành đã được học.
* Các tài liệu về thuật toán FCFS và SJF.
* Tìm kiếm và nghiên cứu trên mạng Internet.
* **Cấu trúc của báo cáo bao gồm:**

**Chương 1:** Tổng quan về vấn đề nghiên cứu.

**Chương 2:** Cài đặt thuật toán.

**Chương 3:** Xây dựng chương trình mô phỏng bằng đồ họa.

**Chương 4:** Kết luận và hướng phát triển.

# TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU

## CƠ SỞ LÝ THUYẾT

### Giới thiệu

#### Khái niệm hệ điều hành

Hệ điều hành là hệ thống phần mềm đóng vai trò trung gian giữa người sử dụng và phần cứng máy tính nhằm tạo ra môi trường giúp thực hiện các chương trình một cách thuận tiện. Ngoài ra, hệ điều hành còn quản lý và đảm bảo cho việc sử dụng phần cứng của máy tính được hiệu quả.

#### Các chức năng chính của hệ điều hành

* Quản lý, chia sẻ tài nguyên
* Quản lý tài nguyên đảm bảo cho tài nguyên hệ thống được sử dụng một cách có ích và hiệu quả. Nhờ có hệ điều hành, tài nguyên được quản lý và sử dụng hợp lý hơn trong khi người sử dụng được giải phóng khỏi công việc khó khăn này.
* Quản lý tài nguyên còn có nghĩa là đảm bảo sao cho chương trình không xâm phạm tài nguyên đã cấp cho chương trình khác. Ví dụ, nếu hai chương trình được cấp hai vùng bộ nhớ khác nhau, thì việc chương trình này truy cập và thay đổi vùng bộ nhớ của chương trình khác sẽ làm cho chương trình đó hoạt động không bình thường. Hệ điều hành cần thể hiện chức năng quản lý tài nguyên của mình qua việc ngăn ngừa những vi phạm kiểu này.
* **Quản lý việc thực hiện các chương trình**
* Hệ điều hành giúp việc chạy chương trình dễ dàng hơn. Để chạy chương trình cần thực hiện một số thao tác nhất định, nhờ có hệ điều hành, người dùng không phải thực hiện các thao tác này. Hệ điều hành cũng cung cấp giao diện giúp người dùng dễ dàng chạy hoặc kết thúc các chương trình.

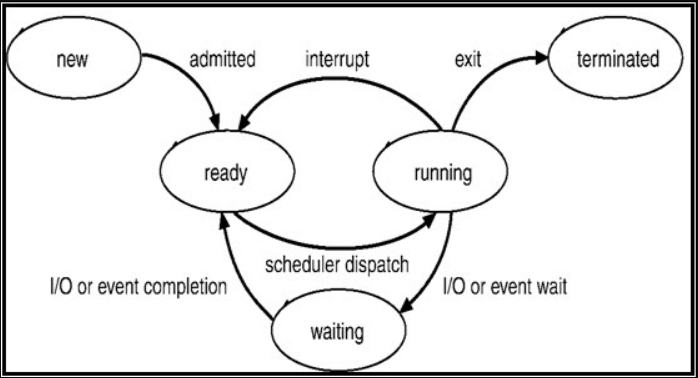
### Các khái niệm cơ bản

#### 1.1.2.1 Khái niệm tiến trình và trạng thái của tiến trình

Tiến trình - process là một khái niệm cơ bản trong bất kì một hệ điều hành nào. Một tiến trình có thể được định nghĩa là một thực thể chương trình đang được chạy trong hệ thống. Một web server chạy trong thiết bị là một tiến trình, hoặc một chương trình soạn thảo văn bản đang chạy trong thiết bị cũng là một tiến trình.

Trạng thái của tiến trình dùng để mô tả tình trạng hiện tại của tiến trình. Là một thực thể động, tiến trình có thể thuộc những trạng thái khác nhau. Trên thực tế, hệ điều hành thường phân biệt năm trạng thái khác nhau của tiến trình: mới, khởi tạo, sẵn sàng, chạy, chờ đợi, kết thúc. Ý nghĩa cụ thể năm trạng thái như sau:

* ***Trạng thái mới khởi tạo (new):*** tiến trình đang được tạo ra. Hệ điều hành đã tạo ra các thông tin về tiến trình tuy nhiên tiến trình chưa được thêm vào danh sách những tiến trình được phép thực hiện. Thông thường, tiến trình ở trạng thái này chưa nằm trong bộ nhớ.
* ***Trạng thái sẵn sàng (ready):*** tiến trình chờ được cấp CPU để thực hiện lệnh của mình.
* ***Trạng thái chạy (running):*** lệnh của tiến trình được CPU thực hiện. Với những máy tính có một CPU và CPU có một lõi, tại mỗi thời điểm chỉ có một tiến trình nằm trong trạng thái chạy.
* ***Trạng thái chờ đợi (waiting):*** tiến trình chờ đợi một sự kiện gì đó xảy ra, ví dụ chờ tín hiệu từ tiến trình khác hoặc chờ kết thúc quá trình vào/ra. Trạng thái chờ đợi còn được gọi là trạng thái bị phong tỏa (blocked).
* ***Trạng thái kết thúc (terminated):*** tiến trình không còn nằm trong danh sách các tiến trình được thực hiện nhưng vẫn chưa bị xóa. Tiến trình thuộc về trạng thái này sau khi đã thực hiện xong hoặc bị tiến trình khác kết thúc.



Hình . Sơ đồ chuyển trạng thái của tiến trình

#### Các đặc điểm của tiến trình

Điều phối hoạt động của các tiến trình là một vấn đề rất phức tạp 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 (I/O-boundedness):*** 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 (CPU-boundedness):*** 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 hỏi 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 đánh 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 đề hoàn tất và rời khỏi hệ thống. Tuy nhiên cũng có quan 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 giảm 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.

#### 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 được gọi là giờ CPU. Tại thời điểm nhất định, chỉ có một tiến trình được phân phối giờ CPU đề hoạt động (thực hiện các lệnh của mình).

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

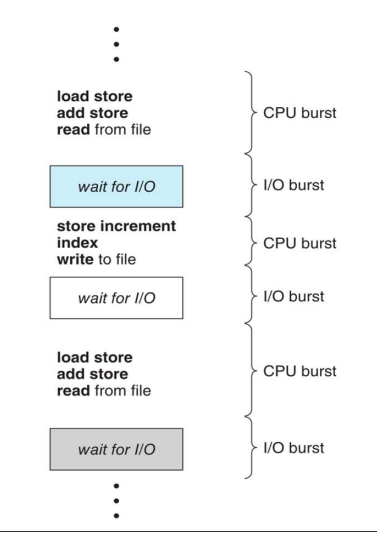
Trong chế độ đa chương trình, có ba trạng thái của tiến trình liên quan mật thiết đến giờ CPU bao gồm:

* ***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 đữ 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.

Một tiến trình đang trong 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 kiện 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 vệc phân phối lại giờ CPU.

#### Chu kì sử dụng CPU I/O

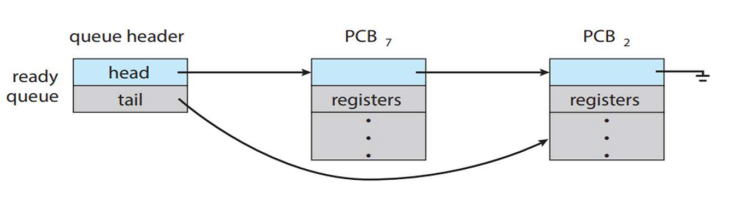


Hình . Chu trình CPU I/O

* CPU Burst là khoảng thời gian mà tiến trình cần xử lý trong CPU. I/O Burst là khoảng thời gian mà tiến trình phải chờ để thực hiện công việc I/O.
* CPU Burst Cycle là toàn bộ khoảng thời gian mà tiến trình cần xử lý trong CPU. I/O Burst Cycle là toàn bộ khoảng thời gian mà tiến trình cần thực hiện công việc I/O.
* Chu kì sử dụng CPU I/O:
* Điều phối CPU thành công phụ thuộc vào việc thực thi tiến trình theo chu kỳ (CPU 🡪 chờ nhập/xuất).
* Chương trình sẽ sử dụng CPU trong một khoảng thời gian.
* Sau đó thi hành thao tác I/O
* Tiếp tục sử dụng CPU, ...

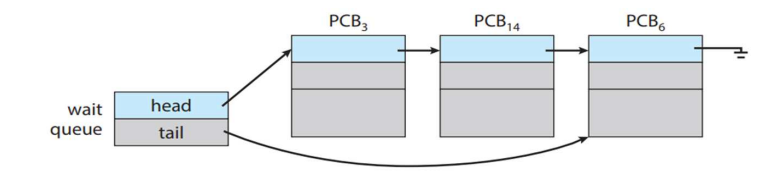
#### Các thông tin cơ bản mà HĐH sử dụng để lập lịch tiến trình

* ***Bậc của đa chương:*** là số lượng các tiến trình đang có mặt trong bộ nhớ chờ được cấp phát CPU để được xử lý chứa trong ready queue có trạng thái ready.
* ***Swapping:*** nếu bậc của đa chương quá lớn thì hệ điều hành chọn các tiến trình có độ ưu tiên thấp chuyển ra ngoài thiết bị lưu trữ, tiến trình có độ ưu tiên cao đưa vào bộ nhớ chính.
* ***I/O-bound process:*** loại tiến trình thiên về trao đổi I/O nhiều hơn
* ***CPU-bound process:*** loại tiến trình thiên về cấp phát CPU
* ***Các hàng đợi***
* Ready queue: chứa thông tin các tiến trình ở trạng thái sẵn sàng và chỉ chờ cấp phát CPU đã hoàn thiện việc cấp phát đầy đủ tài nguyên.



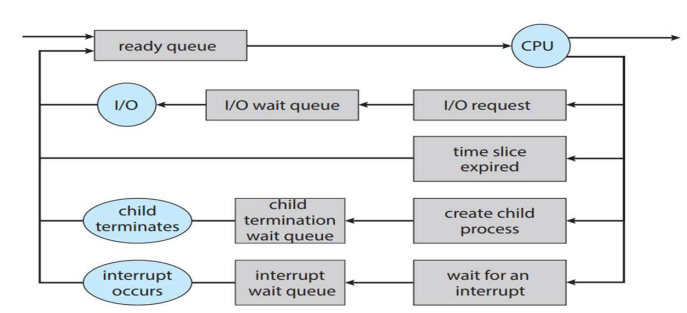
Hình . Ready queue

* Wait queue: Chứa thông tin các tiến trình công việc I/O (chứa thông tin các tiến trình đang ở trang thái waiting).



Hình . Wait queue

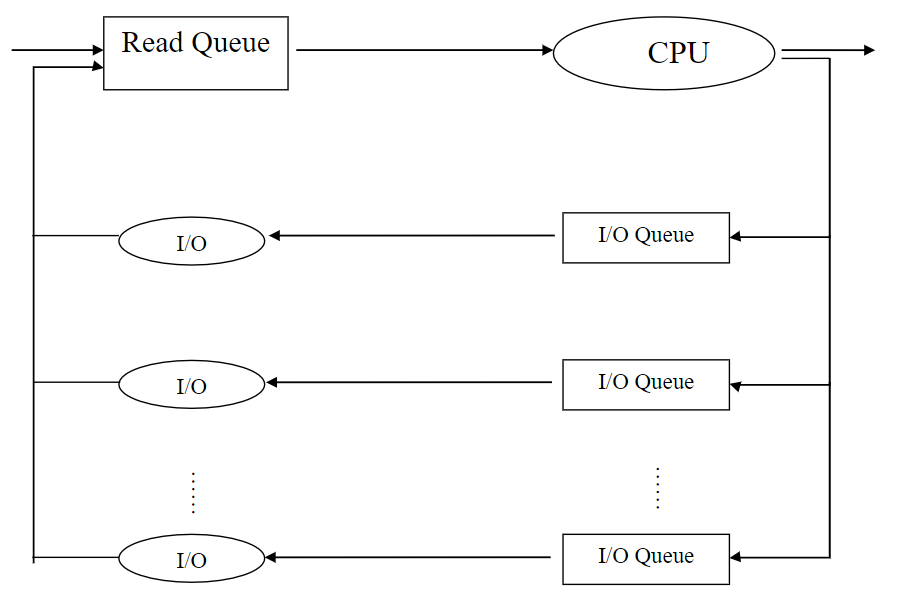
* Child termination wait queue: Chứa thông tin các tiến trình mà tiến trình đó phải chờ các tiến trình con kết thúc.
* Interrupt queue: chứa thông tin các tiến trình bị gián đoạn do một vấn đề nào đó đột ngột phát sinh.



* ***Context switch*** (kỹ thuật chuyển ngữ cảnh). Đây là cơ chế lưu trữ và khôi phục trạng thái hoặc ngữ cảnh của CPU trong PCB (Process Control Block để lưu dữ liệu hoạt động của các tiến trình). Nó cho phép thực hiện tiến trình được bắt đầu lại từ cùng một trạng thái kết thúc trước đó. Context switch là một tính năng thiết yếu của hệ điều hành đa nhiệm.

#### Khái niệm lập lịch cho CPU và lý do cần phải lập lịch 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) đề 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:



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, quyết định tiến trình nào được sử dụng tài nguyên phần cứng khi nào, trong thời gian bao lâu. Bài toán điều độ được đặt ra với mọi dạng tài nguyên khác nhau, chẳng hạn thiết bị vào ra, CPU, bộ nhớ..., kể cả trong trường hợp có chia sẻ thời gian hay không.

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...

Lý do cần phải lập lịch CPU: trong hệ thống multitasking (đa nhiệm) thì tại mỗi thời điểm trong bộ nhớ có nhiều process (tiến trình) nhưng tại mỗi thời điểm chỉ có một process được thực thi do đó cần phải giải quyết vấn đề phân chia, do đó cần lập lịch để phân phối thời gian sử dụng CPU cho các tiến trình của người dụng và sử hệ thống.

#### 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 lượt khác nhau để thực hiện việc điều phối, nhưng tựu chung cần đạt được các mục tiêu sau:

* ***Sự công bằng (Fairness):*** các tiến trình chia sẻ CPU một cách công bằng không có tiến trình nào phải đợi vô hạn để được cấp phát CPU.
* ***Tính hiệu quả (Efficiency):*** Chúng ta muốn giữ CPU bận nhiều nhất có thể. Việc sử dụng CPU có thể từ 0 đến 100%. Trong hệ thống thực, nó nên nằm trong khoảng từ 40% (cho hệ thống được nạp tải nhẹ) tới 90% (cho hệ thống được nạp tải nặng).
* ***Thời gian chờ nhỏ nhất (Waiting time):*** cực tiểu hóa tổng thời gian chờ của tiến trình trong hàng đợi sẵn sàng.
* ***Thời gian đáp ứng hợp lý (Response time):*** cực tiểu hóa thời gian hỏi đáp cho các tương tác của người sử dụng, thời gian từ lúc gởi yêu cầu cho tới khi đáp ứng đầu tiên được tạo ra.
* ***Thời gian hoàn thành nhỏ nhất (Turnaround Time):*** cực tiểu hóa khoảng thời gian từ thời điểm khởi tạo tiến trình tới khi tiến trình hoàn thành các tác vụ xử lý theo lô.
* ***Thông lượng tối đa (Throughput):*** cực đại hóa số lượng tiến trình được hoàn thành trên một đơn vị thời gian.

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

#### Điều phối không độc quyền (Nonpreemptive scheduling) và điều phối độc quyền (Preemptive scheduling)

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 lý 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.

Các giải thuật độc quyền thường đơn giản và dễ cài đặt. Tuy nhiên chúng thường không thích hợp với các hệ thống tổng quát nhiều người dùng, vì nếu cho phép một tiến trình có quyền xử lý bao lâu tùy ý, có nghĩa là tiến trình này đã giữ CPU một thời gian không xác định, có thể ngăn cản những tiến trình còn lại trong hệ thống có một cơ hội để xử lý.

* 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 có 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 thế bị tạm dừng hoạt động bất cứ lúc nào mà không được báo trước để tiến trình khác xử lý. Các quyết định điều phối xảy ra khi:

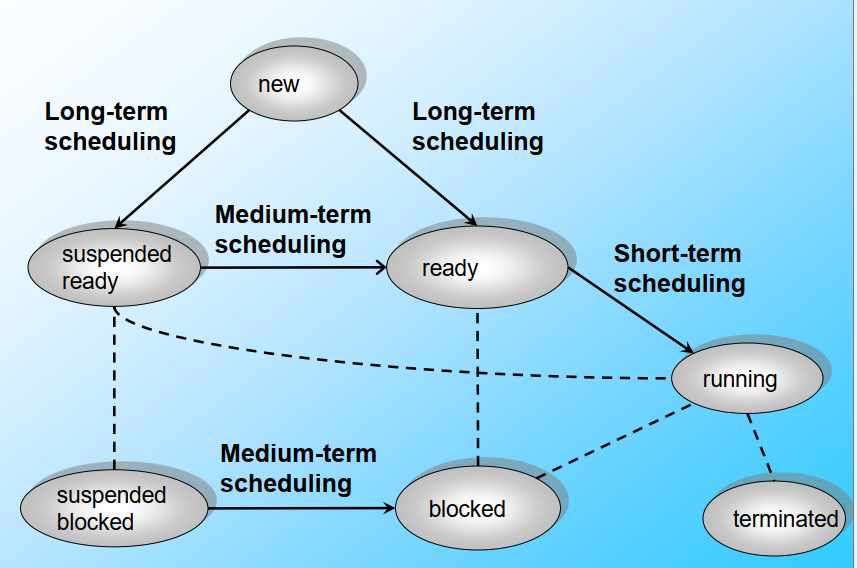
* Khi tiến trình chuyển từ trạng thái đang xử lý (running) sang trạng thái bị khóa blocked.
* Khi tiến trình chuyển từ trạng thái đang xử lý (running) sang trạng thái ready (vì xảy ra một ngắt).
* Khi tiến trình chuyển từ trạng thái chờ (blocked) sang trạng thái ready (ví dụ một thao tác nhập xuất hoàn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

Trong các hệ thống sử dụng nguyên 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 (realtime), cần phải sử dụng nguyên lý điều phối không độc quyền đề các tiến trình quan trọng có cơ hội hồi đáp kịp thời. Tuy nhiên thực hiện hiện điều phối theo nguyên lý không độc quyên đòi hỏi nhưng cơ chế phức tạp trong việc phân định độ ưu tiên, và phát sinh thêm chi phí khi chuyển đổi CPU qua lại giữa các tiến trình.

#### Các bộ điều phối

* Long-term scheduling
* Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi.
* Điều khiển mức độ multiprogramming của hệ thống.
* Long term scheduler thường cố gắng duy trì xen lẫn CPU-bound và I/O-bound process.
* Medium-term scheduling
* Process nào được đưa vào (swap in), đưa ra khỏi (swap out) bộ nhớ chính
* Được thực hiện bởi phần quản lý bộ nhớ và được thảo luận ở phần quản lý bộ nhớ.
* Short term scheduling
* Xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp (còn được gọi là định thời CPU, CPU scheduling)
* Short term scheduler còn được gọi với tên khác là dispatcher
* Bộ định thời short-term được gọi mỗi khi có một trong các sự kiện/interrupt sau xảy ra:
* Ngắt thời gian (clock interrupt)
* Ngắt ngoại vi (I/O interrupt)
* Lời gọi hệ thống (operating system call)
* Signal



#### Khái niệm Dispatcher (bộ điều phối)

Dispatcher là 1 bộ phận, 1 chức năng của bộ định thời ngắn hạn

Nhiệm vụ: sau khi CPU scheduler đã chọn được 1 tiến trình để giao CPU rồi, Dispatcher sẽ thực hiện công việc tiếp theo, thực hiện những công việc cần thiết gồm:

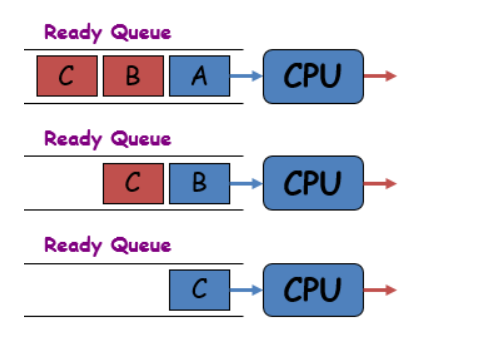
* Chuyển đổi ngữ cảnh từ tiến trình cũ đang giữ CPU sang tiến trình mới sẽ được nhận CPU
* Chuyển chế độ thực hiện sang user mode
* Nhảy đến vùng nhớ chứa chỉ thị lệnh tiếp theo của tiến trình được chọn để giao cho CPU xử lý

***Dispatch latency:*** khoảng thời gian Dispatcher chuyển đổi ngữ cảnh từ tiến trình cũ giữ CPU sang tiến trình mới. Dispatcher phải giảm số lần chuyển đổi ngữ cảnh để Dispatch latency càng nhỏ cảng tốt.

### Các giải thuật điều phối CPU cơ bản

#### First Come First Served (FCFS)

Cho đến nay, thuật toán định thời CPU đơn giản nhất là first-come, first-server (FCFS). Hiểu đơn giản nghĩa của thuật toán này là: đến trước, phục vụ trước. Với giải thuật này, nó ứng dụng chế độ nonpreemptive – tức là một tiến trình sẽ chiếm giữ CPU cho đến khi nào nó trả CPU, thông qua phương thức ngắt hay yêu cầu nhập/xuất xuất hiện.



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.

* Ưu điểm
* Giờ CPU không bị phân phối lại (không bị ngắt).
* 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

Do thời gian trung bình chờ phục vụ của các tiến trình là như nhau (không kể tiến trình ngắn hay dài), do đó dẫn tới ba điểm sau:

* 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.
* Ứng dụng của giải thuật FCFS:

FCFS thường được sử dụng trong các hệ thống bó (batch system). Giải thuật này đặc biệt không phù hợp với các hệ phân chia thời gian, trong các hệ này, cần cho phép mỗi tiến trình được cấp phát CPU đều đặn trong từng khoảng thời gian.

* Ví dụ:

**Ví dụ 1:**

Process: Công việc cần xử lý.

Burst time: Thời gian cần thiết để hoàn thành công việc.

|  |  |
| --- | --- |
| Process | Burst Time |
| P1 | 24 |
| P2 | 3 |
| P3 | 3 |

**Trường hợp 1**: 3 tiến trình vào hàng đợi theo thứ tự: P1 , P2 , P3.

***Sơ đồ Gantt:***

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

0 24 27 30

* Thời gian chờ P1 = 0 ms; P2 = 24 ms; P3 = 27 ms.
* Thời gian chờ trung bình: (0 + 24 + 27)/3 = 17 ms.
* Thời gian hoàn thành trung bình: (24 + 27 + 30)/3 = 27 ms.

Điểm yếu: Các tiến trình có thời gian CPU ngắn vào sau tiến trình có thời gian CPU dài.

**Trường hợp 2**: Giả sử vào hàng đợi theo thứ tự: P2 , P3 , P1.

***Sơ đồ Gantt:***

|  |  |  |
| --- | --- | --- |
| P2 | P3 | P1 |

0 3 6 30

* Thời gian chờ P1 = 6 ms; P2 = 0 ms; P3 = 3 ms.
* Thời gian chờ trung bình: (6 + 0 + 3)/3 = 3 ms.
* Thời gian hoàn thành trung bình: (3 + 6 + 30)/3 = 13 ms.

***Nhận xét:***

* Thời gian chờ trung bình tốt hơn (3 < 17).
* Thời gian hoàn thành trung bình tốt hơn (13 < 27).

**Ví dụ 2:**

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 24 |
| P2 | 1 | 3 |
| P3 | 2 | 3 |

Arrival time: Thời gian đến của tiến trình

Nếu áp dụng FCFS để lập lịch trình cho bộ xử lý thì P1 đến đầu tiên cho nên nó sẽ được thực thi đầu tiên. Sau khi P0 hoàn tất mất khoảng 24ms, tại thời điểm này hàng đợi đã có cả P2 và P3. Tuy nhiên P2 đến sớm hơn cho nên nó sẽ được xử lý trước. Sau đó P3 sẽ được thực thi cuối cùng.

***Sơ đồ Gantt:***

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

0 24 27 30

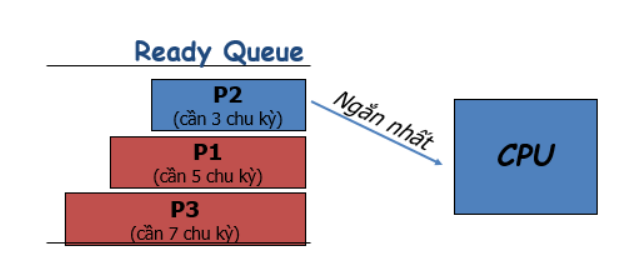
* Thời gian chờ của P0=0 ms, P1= 24 – 1= 23 ms, P2= 27 – 2= 25 ms.
* Thời gian chờ trung bình là (0+23+25)/3 = 16 ms.
* Thời gian hoàn thành (Turnaround Time = thời gian kết thúc – thời gian đến).

|  |  |
| --- | --- |
| Process | Turnaround Time |
| P0 | 24 – 0 = 24 |
| P1 | 27 – 1 = 26 |
| P2 | 30 – 2 = 28 |

Thời gian hoàn thành trung bình = (24+26+28)/3 = 26ms.

#### Shortest Job First (SJF)

Một tiếp 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 biểu 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 tốn nhiều xử lý cho quá trình quản lý.
* Mặc dù SJF là tối ưu nhưng nó không thể 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 giải thuật này có nhiều dị bản khác nhau và sẽ tối ưu hay không tối ưu phụ thuộc vào trưng dụng CPU. Giải thuật SJF không độc quyền sẽ dừng hoạt động của tiến trình hiện hành, trong khi giải thuật độc quyền sẽ cho phép tiến trình hiện hành tiếp tục xử lý.
* Ví dụ:

Ví dụ 1: Không có thời gian đến của tiến trình

|  |  |
| --- | --- |
| Process | Burst Time |
| P1 | 6 |
| P2 | 8 |
| P3 | 7 |
| P4 | 3 |

Burst time của P4 là nhỏ nhất nên được giữ CPU để xử lý trước sau đó đến P1, P3 và cuối cùng là P2 được xử lý.

***Sơ đồ Gantt:***

|  |  |  |  |
| --- | --- | --- | --- |
| P4 | P1 | P3 | P2 |

0 3 9 16 24

* Thời gian chờ đợi: P1 = 3 ms, P2 = 16 ms, P3 = 9 ms, P4 = 0 ms.
* Thời gian chờ trung bình: (3 + 6 + 9 + 0)/4 = 7 ms.
* Thời gian hoàn thành trung bình: (3 + 9 + 16 + 24)/4 = 13 ms.

Ví dụ 2: Có thời gian đến của tiến trình

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | Burst Time |
| P1 | 0 | 6 |
| P2 | 1 | 8 |
| P3 | 2 | 7 |
| P4 | 3 | 3 |

Arrival time: Thời gian đến của tiến trình.

P1 đến đầu tiên cho nên nó sẽ được thực thi đầu tiên và giữ CPU trong 6ms vì là điểu phối theo cơ chế độc quyền. Sau khi P1 hoàn tất, tại thời điểm này hàng đợi đã có 3 tiến trình P2, P3 và P4. Vì P4 có Burst time nhỏ nhất trong 3 tiến trình nên được cấp phát CPU để xử lý trước. Sau đó P3 sẽ được thực thi rồi tới P2 (vì 7 < 8).

***Sơ đồ Gantt:***

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P4 | P3 | P2 |

0 6 9 16 24

* Thời gian chờ của P1=0 ms, P2= 16 – 1= 15 ms, P3= 9 – 2= 7 ms, P4=6–3= 3 ms
* Thời gian chờ trung bình là (0 + 15 + 7 + 3)/4 = 6.25 ms.
* Thời gian hoàn thành (Turnaround Time = thời gian kết thúc – thời gian đến).

|  |  |
| --- | --- |
| Process | Turnaround Time |
| P1 | 6 – 0 = 6 |
| P2 | 24 – 1 = 23 |
| P3 | 16 – 2 = 14 |
| P4 | 9 – 3 = 6 |

Thời gian hoàn thành trung bình = (6 + 23+ 14 + 6)/4 = 12.25ms.

## CÁC NGÔN NGỮ ĐƯỢC SỬ DỤNG

### Ngôn ngữ C# (C-Sharp)

#### Khái niệm

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth.

C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

#### Nền tảng .NET Framework

.NET Framework là một thư việc class có thể được sử dụng với một ngôn ngữ .NET để thực thi việc thao tác chuỗi cho đến phát sinh ra các web động (ASP.NET), phân tích XML và reflection. Framework được tổ chức thành các tập hợp namespace để nhóm các class có cùng chức năng với nhau.

Ví dụ:

*System.Drawing* cho đồ họa.

*System.Collection* cho cấu trúc dữ liệu.

*System.Windows.Forms* cho hệ thống Windows Forms.

#### Ứng dụng của C Sharp

C# là một ngôn ngữ lập trình có mục đích chung để xây dựng các loại chương trình và ứng dụng khác nhau. C# có tính linh hoạt rất cao, tuy nhiên có ba lĩnh vực nó thường được sử dụng:

* Phát triển ứng dụng web

Ngôn ngữ lập trình C# có thể được sử dụng trên bất kỳ nền tảng nào hiện nay. Bạn có thể xây dựng các trang web động và ứng dụng web với nền tảng .NET hay các nền tảng open-source khác. C# có thể làm cho ứng dụng web chạy trơn chu trên một máy chủ.

* Ứng dụng Windows

Microsoft đã tạo ra C# cho chính Microsoft. Vì vậy, không cần nghi ngờ vì sao C# lại rất phổ biến trong việc xây dựng các ứng dụng Windows.

Ngoài ra, các developer có thể tin tưởng vào sự hỗ trợ của cộng đồng cùng với các tài liệu về phát triển ứng dụng, chương trình cụ thể cho kiến trúc của nền tảng Microsoft.

* Lập trình Games

Trong thế giới game, các developer thường ưa thích việc sử dụng ngôn ngữ lập trình C Sharp (C#). Ngôn ngữ này đặc biệt mạnh mẽ trong việc xây dựng các trò chơi “hot” hiện nay. Ngoài ra, game engine Unity – một trong những game engine phổ biến nhất hiện nay – cũng được xây dựng bằng C++ và C#.

#### Class trong lập trình hướng đối tượng và C#

Class trong các ngôn ngữ lập trình hướng đối tượng nói chung, trong C# nói riêng, mang ý nghĩa là một kiểu dữ liệu. Đối với C#, class là các khối xây dựng cơ sở của các chương trình ứng dụng và là trung tâm của lập trình C#.

Class trong C# là một loại kiểu dữ liệu đặc biệt chứa định nghĩa những thuộc tính (thông tin) và phương thức (hành vi), dùng để mô tả chung cho một nhóm những thực thể cùng loại.

Trong C#, mỗi class có thể chứa:

* ***Biến thành viên (field):*** Lưu trữ các thông tin mô tả về đối tượng hay trạng thái của đối tượng.
* ***Thuộc tính (property):*** có vai trò lưu trữ thông tin tương tự như biến thành viên nhưng có khả năng kiểm soát dữ liệu xuất nhập.
* ***Phương thức (method):*** Dùng để cập nhật, tính toán, cung cấp và xử lý thông tin.
* ***Sự kiện (delegate/event):*** Gửi thông báo về sự thay đổi trạng thái của đối tượng ra bên ngoài.

Ngoài ra, trong class còn có thể chứa định nghĩa của kiểu dữ liệu khác, gọi là kiểu thành viên (member /inner/nested type). Class có thể chứa định nghĩa của bất kỳ nhóm kiểu nào mà bạn đã biết (class, struct, interface, delegate, enum).

Khi xem class như một kiểu dữ liệu thì object của class tương ứng chính là biến thuộc kiểu dữ liệu đó. Class chứa mô tả trừu tượng còn object chứa giá trị cụ thể của mỗi mô tả đó.

#### Ưu điểm của ngôn ngữ C Sharp là gì?

Lập trình C# là ngôn ngữ phổ biến nhất trong số các ngôn ngữ của Microsoft. Dưới đây là một số thống kê chứng minh cho điều đó:

Dựa theo cuộc khảo sát của StackOverflow vào năm 2019, có đến 31.9% các developer chuyên nghiệp ưa thích ngôn ngữ này. Ngoài ra, cộng đồng C# cũng lớn thứ ba trong cùng cuộc khảo sát.

Nhận được 67% sự bình chọn cho ngôn ngữ lập trình được yêu thích nhất thế giới. Hơn 5000 công việc C# được quảng cáo, tính riêng ở Hoa Kỳ. Ngoài ra, những đặc điểm vượt trội cũng không dừng lại ở đó:

* C Sharp là ngôn ngữ đơn giản

C# đã được loại bỏ sự phức tạp như các ngôn ngữ C++ và Java bao gồm loại bỏ macro, template, lớp cơ sở (virtual base class). Nếu bạn đã sử dụng qua các ngôn ngữ C, C++ hay cả Java thì C# cũng nó cũng tương tự về cú pháp, biểu thức, nhưng nó được cải tiến để làm ngôn ngữ đơn giản hơn.

* Ngôn ngữ hướng đối tượng

Lập trình hướng đối tượng (OOP – Object Oriented Programming) là phương pháp lập trình với 4 tính chất:

* Tính trừu tượng (abstraction).
* Tính đóng gói (encapsulation).
* Tính đa hình (polymorphism).
* Tính kế thừa (inheritance).
* Thời gian develop nhanh hơn

Ưu thế lớn nhất của chính là khả năng tiết kiệm thời gian develop, vì:

* Được nhập tĩnh, dễ đọc, giảm thiểu thời gian debug.
* Sở hữu thư viện khổng lồ, cung cấp các chức năng cấp cao so với các ngôn ngữ khác như Java hay C++.
* Mang lại sự đơn giản và hiệu quả. Đồng thời còn hỗ trợ các lập trình viên viết các đoạn code phức tạp.
* Có ngân hàng bộ nhớ mở rộng, giảm thời gian develop.
* Đường cong học tập thấp

Các developer, đặc biệt là full-stack developer, có xu hướng thích ngôn ngữ C# hơn. Ngoài việc tiết kiệm thời gian develop, nó còn có đường cong học tập (learning curve) thấp. Các developer có thể dành ít thời gian hơn cho việc học C# so với các ngôn ngữ khác. Tính đơn giản và dễ sử dụng rất thuận lợi cho cả những developer mới.

* Khả năng mở rộng cao

Phát triển phần mềm yêu cầu ngôn ngữ lập trình dễ dàng bảo trì và có thể mở rộng và C# là một ngôn ngữ như vậy. Tính nghiêm ngặt của mã hóa tĩnh làm cho các chương trình nhất quán với nhau. Điều này cho phép các developer dễ dàng thực hiện các điều chỉnh cũng như bảo trì.

Ngoài ra, đối với các developer làm việc với những dự án cũ được viết bằng C#, hầu như sẽ không có thay đổi đáng kể nào với dự án của họ.

* Sự hỗ trợ cộng đồng lớn

Kể từ khi gã khổng lồ trong công nghệ Microsoft tạo ra C#, khó có thể bàn cãi về việc nó nhận được sự hỗ trợ rất lớn từ cộng đồng. Trong thế giới phát triển phần mềm, việc có một cộng đồng là vô cùng cần thiết. Đó là nơi các developer có thể tìm kiếm trợ giúp cũng như các thông tin chi tiết. Ngôn ngữ lập trình không phải là một nền tảng hay dịch vụ có đường dây trợ giúp chuyên dụng, hay những người trợ giúp về CNTT.

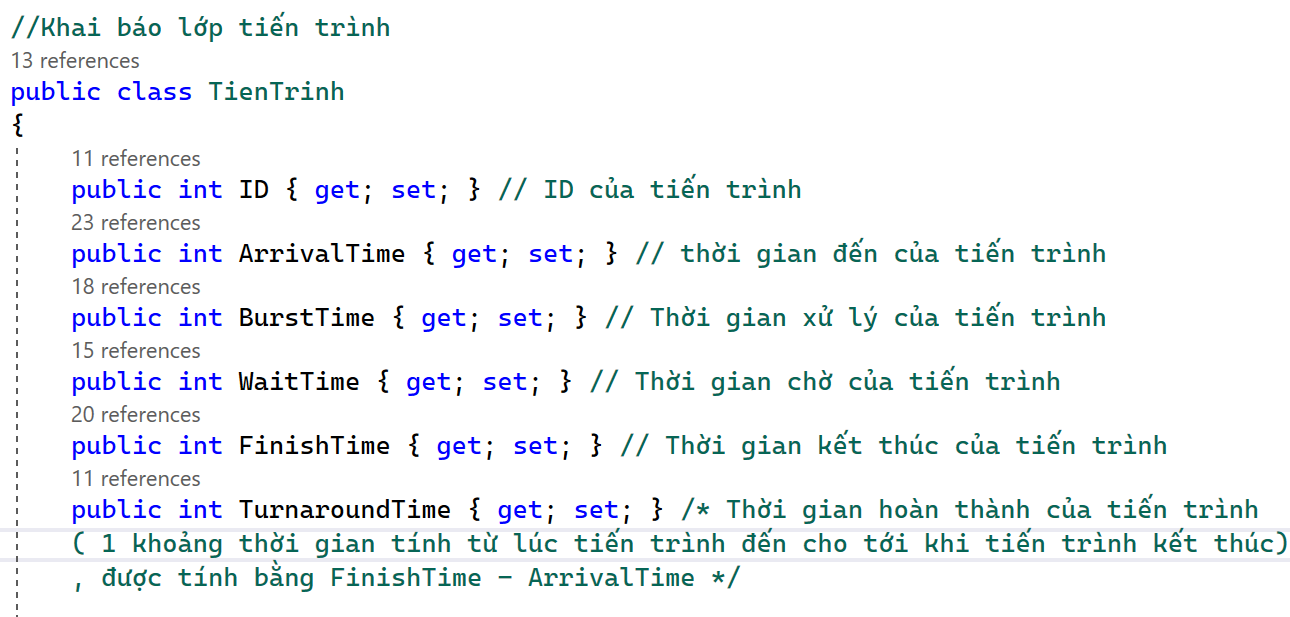
Trong cộng đồng, các developer có thể dựa vào những người có chung lĩnh vực quan tâm. Họ có thể tìm thấy câu trả lời và giải pháp từ những người đã từng trải. Hiện nay, có thể kể đến StackOverflow – một trong những cộng đồng lớn nhất về code.

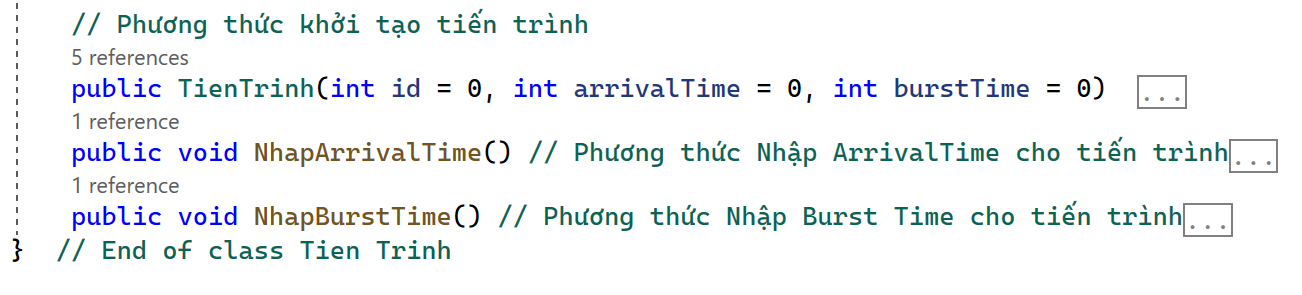
# CÀI ĐẶT THUẬT TOÁN

## Cấu trúc chương trình

### Class Tiến trình

Dùng Class để quản lý thông tin của tiến trình





* Các thuộc tính của Class TienTrinh
* ID: Định danh cho tiến trình đó, mỗi tiến trình sẽ được gán 1 ID để phân biệt với các tiến trình khác.
* ArrivalTime: thời gian đến của tiến trình
* BurstTime: thời gian xử lý của tiến trình
* WaitTime: thời gian chờ của tiến trình
* FinishTime: mốc thời gian kết thúc của tiến trình
* TurnaroundTime: Thời gian hoàn thành của tiến trình (1 khoảng thời gian tính từ lúc tiến trình đến cho tới khi tiến trình kết thúc) , được tính bằng FinishTime - ArrivalTime
* Các phương thức của Class TienTrinh

***Phương thức khởi tạo tiến trình***

|  |
| --- |
| public TienTrinh(int id = 0, int arrivalTime = 0, int burstTime = 0)  {  ID = id;  ArrivalTime = arrivalTime;  BurstTime = burstTime;  } |

***Phương thức nhập thời gian đến cho tiến trình***

|  |
| --- |
| public void NhapArrivalTime() // Phương thức Nhập ArrivalTime cho tiến trình  {  bool kt = false;  do  { // Xử lý ngoại lệ nếu không nhập giá trị số  try  {  //Console.Write("Nhap Arrival Time: ");  ArrivalTime = int.Parse(Console.ReadLine());  kt = true;  if (ArrivalTime < 0)  {  Console.WriteLine("Vui long nhap lai Arrival Time  (ArrivalTime>=0)");  }  }  catch (Exception)  {  Console.WriteLine("Vui long nhap gia tri so!");  }  //Kiểm tra nếu giá trị nhập vào không phải số hoặc ArrivalTime < 0  thì nhập lại  } while (kt == false || ArrivalTime < 0);  } |

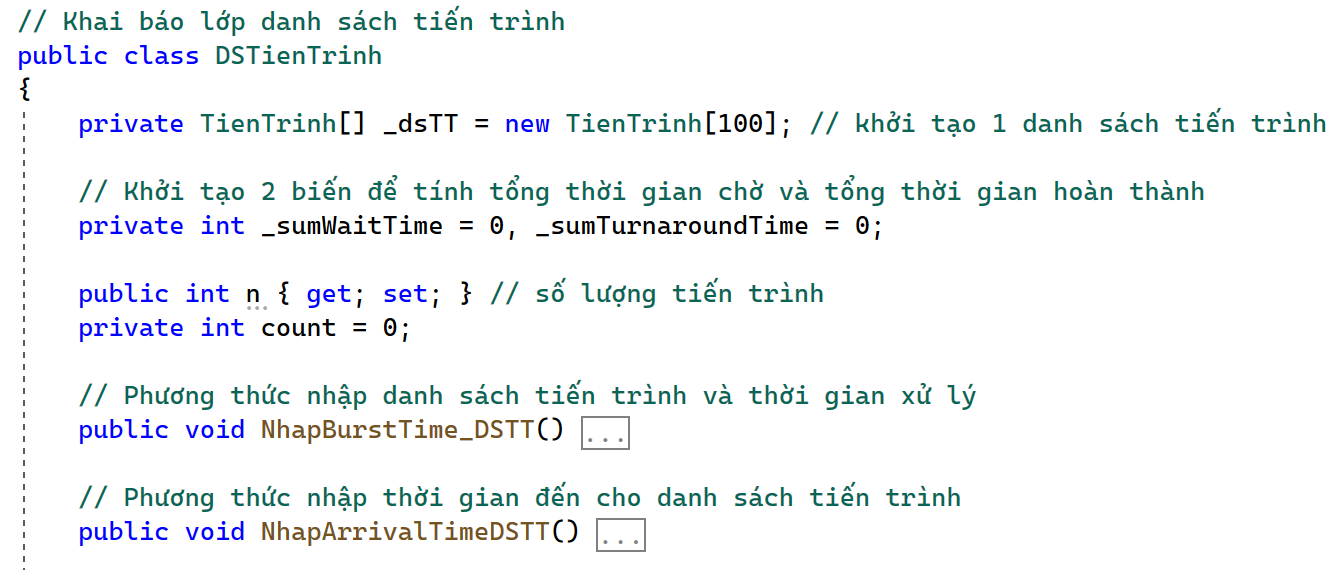
***Phương thức nhập thời gian xử lý cho tiến trình***

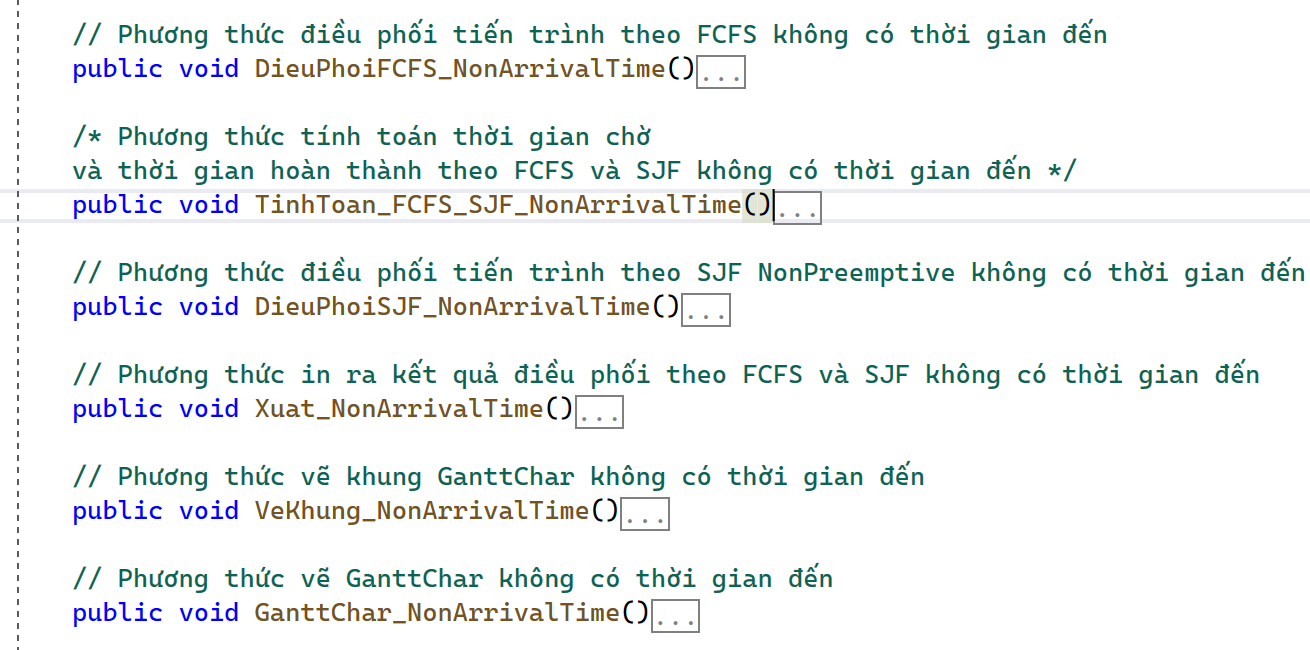
|  |
| --- |
| public void NhapBurstTime() // Phương thức Nhập Burst Time cho tiến trình  {  bool kt = false;  do  { // Xử lý ngoại lệ nếu không nhập giá trị số  try  {  //Console.Write("Nhap Burst Time: ");  BurstTime = int.Parse(Console.ReadLine());  kt = true;  if (BurstTime < 0)  {  Console.WriteLine("Vui long nhap lai Burst Time (Burst Time>=0)  ");  }  }  catch (Exception)  {  Console.WriteLine("Vui long nhap gia tri so!");  }  // Kiểm tra nếu giá trị nhập vào không phải số hoặc BurstTime < 0  thì nhập lại  } while (kt == false || BurstTime < 0);  } |

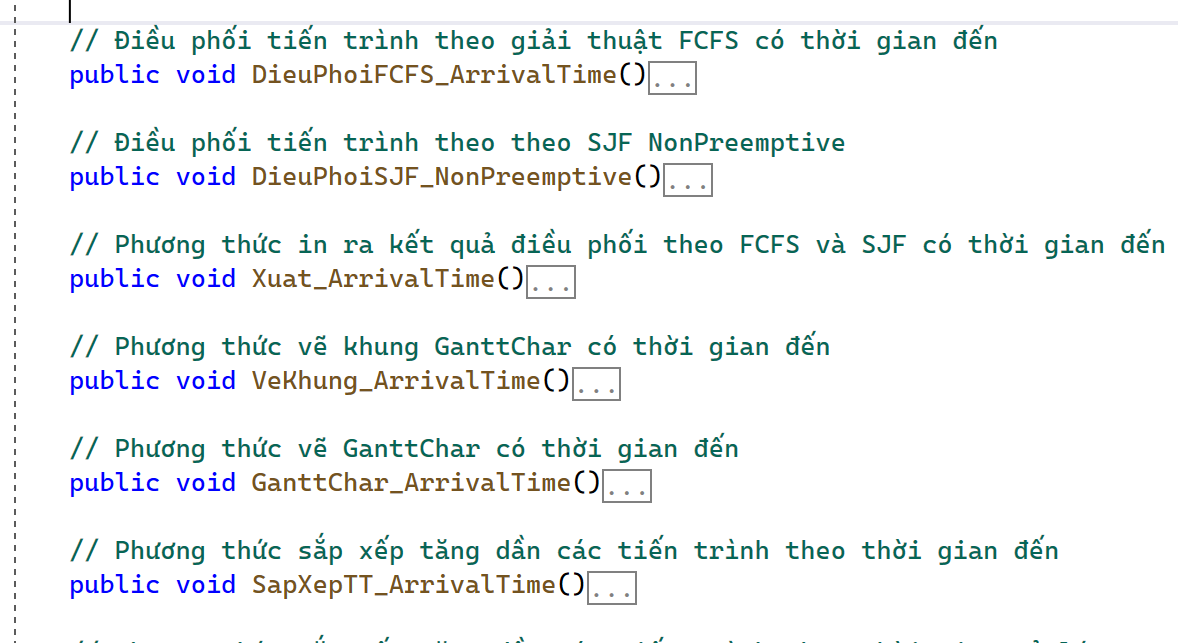
Từ khóa public để các lớp khác cùng namespace có thể truy xuất được

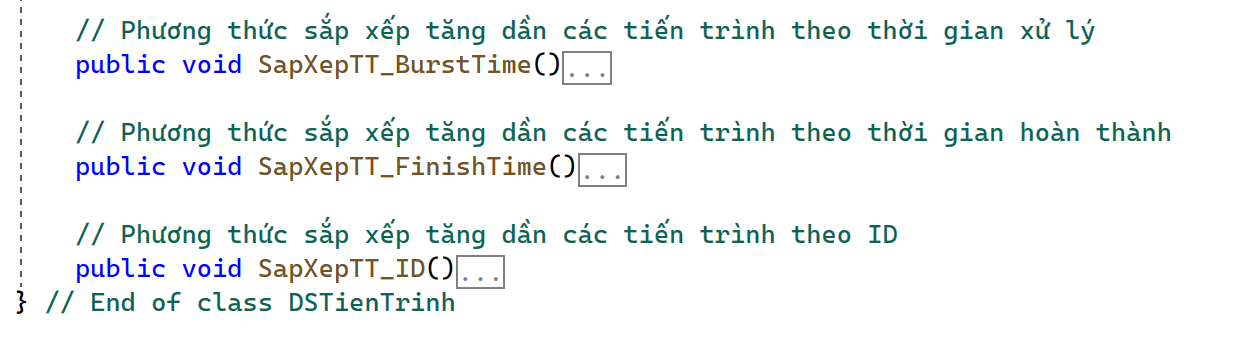
### Class danh sách tiến trình

Tạo 1 Class DSTienTrinh để tạo điều kiện thuận lợi cho việc quản lý, điều phối các tiến trình.









* Các thuộc tính của Class DSTienTrinh
* \_dsTT: khởi tạo 1 danh sách tiến trình
* \_sumWaitTime: tổng thời gian chờ
* \_sumTurnaroundTime: tổng thời gian hoàn thành
* n: số lượng tiến trình
* count: biến đếm
* Các phương thức của Class DSTienTrinh

***Phương thức nhập danh sách tiến trình và thời gian xử lý:***

|  |
| --- |
| // Phương thức nhập danh sách tiến trình và thời gian xử lý  public void NhapBurstTime\_DSTT()  {  do  {  try  {  Console.Write("Nhap so luong tien trinh (n>0): ");  n = int.Parse(Console.ReadLine());  if (n <= 0)  {  Console.WriteLine("Vui long nhap so luong lon hon 0");  }  }  catch (Exception)  {  Console.WriteLine("Vui long nhap gia tri so!");  }  } while (n <= 0); // Nếu n<=0 thì nhập lại  \_dsTT = new TienTrinh[n]; // đinh nghĩa 1 danh sách gồm n tiến trình  for (int i = 0; i < n; i++) // Nhập thông tin từng tiến trình  {  \_dsTT[i] = new TienTrinh();  Console.Write("Nhap Burst Time cho tien trinh P{0} : ", i + 1);  \_dsTT[i].NhapBurstTime();  \_dsTT[i].ID = i + 1;  }  } |

***Phương thức nhập thời gian đến cho danh sách tiến trình***

|  |
| --- |
| // Phương thức nhập thời gian đến cho danh sách tiến trình  public void NhapArrivalTimeDSTT()  {  for (int i = 0; i < n; i++) // Nhập thông tin từng tiến trình  {  Console.Write("Nhap Arrival Time cho tien trinh P{0} : ", i + 1);  \_dsTT[i].NhapArrivalTime();  }  } |

***Phương thức điều phối tiến trình theo FCFS không có thời gian đến***

|  |
| --- |
| // Phương thức điều phối tiến trình theo FCFS không có thời gian đến  public void DieuPhoiFCFS\_NonArrivalTime()  {  SapXepTT\_ID(); // Sắp xếp lại thứ tự danh sách theo ID (là thứ tự vào hàng đợi)  Console.WriteLine("\n\t\t\t ----- GIAI THUAT FIRST COME FIRST SERVED (Non ArrivalTime) ----- ");  TinhToan\_FCFS\_SJF\_NonArrivalTime();  } |

***Phương thức tính toán thời gian chờ và thời gian hoàn thành theo FCFS và SJF không có thời gian đến***

|  |
| --- |
| /\* Phương thức tính toán thời gian chờ và thời gian hoàn thành theo FCFS và SJF không có thời gian đến \*/  public void TinhToan\_FCFS\_SJF\_NonArrivalTime()  {  \_sumWaitTime = 0; \_sumTurnaroundTime = 0;  // Tiến trình đầu tiên vào danh sách nên thời gian chờ = 0  \_dsTT[0].WaitTime = 0;  // Tính thời gian chờ của các tiến trình còn lại  for (int i = 1; i < n; i++)  {  /\* Thời gian chờ của tiến trình = thời gian chờ đợi của tiến trình  trước + thời gian xử lý của tiến trình trước \*/  \_dsTT[i].WaitTime = \_dsTT[i - 1].WaitTime + \_dsTT[i - 1].BurstTime;  }  /\* Tính thời gian hoàn thành của các tiến trình còn lại và tổng thời gian  chờ tổng thời gian hoàn thành \*/  for (int i = 0; i < n; i++)  {  /\* Thời gian hoàn thành của tiến trình = thời gian chờ đợi + thời gian  xử lý của tiến trình /\*  \_dsTT[i].TurnaroundTime = \_dsTT[i].WaitTime + \_dsTT[i].BurstTime;  \_sumWaitTime += \_dsTT[i].WaitTime;  \_sumTurnaroundTime += \_dsTT[i].TurnaroundTime;  }  } |

***Phương thức điều phối tiến trình theo SJF NonPreemptive không có thời gian đến***

|  |
| --- |
| // Phương thức điều phối tiến trình theo SJF NonPreemptive không có thời gian đến  public void DieuPhoiSJF\_NonArrivalTime()  {  SapXepTT\_ID();  SapXepTT\_BurstTime();//Sắp xếp danh sách tiến trình tăng dần theo BurstTime  Console.WriteLine("\n\t\t\t ----- GIAI THUAT SHORTEST FIRST JOB (Non  ArrivalTime) ----- ");  TinhToan\_FCFS\_SJF\_NonArrivalTime();  } |

***Phương thức in ra kết quả điều phối theo FCFS và SJF không có thời gian đến***

|  |
| --- |
| // Phương thức in ra kết quả điều phối theo FCFS và SJF không có thời gian đến  public void Xuat\_NonArrivalTime()  {  // Vẽ sơ đồ gantt  GanttChar\_NonArrivalTime();  // In ra thông tin của các tiến trình cũng như thứ tự được xử lý của chúng  Console.WriteLine("\nBang thong tin cua cac tien trinh\n");  Console.WriteLine("| Tien trinh | Thoi gian xu ly | Thoi gian cho | Thoi  gian hoan thanh |");  for (int i = 0; i < n; i++)  {  Console.WriteLine("|\tP{0}\t|\t {1}\t |\t {2}\t |\t {3} \t  |\t", \_dsTT[i].ID, \_dsTT[i].BurstTime, \_dsTT[i].WaitTime,  \_dsTT[i].TurnaroundTime);  }  // In ra thời gian chờ trung bình và thời gian hoàn thành trung bình  Console.WriteLine("\n Thoi gian cho trung binh: {0:0.0.0} ms", \_sumWaitTime  \* 1.0 / n);  Console.WriteLine(" Thoi gian hoan thanh trung binh: {0:0.0.0} ms",  \_sumTurnaroundTime \* 1.0 / n);  Console.WriteLine("--------------------------------------------------------  ------------------------------");  } |

***Phương thức vẽ khung GanttChar và vẽ GanntChar không có thời gian đến***

|  |
| --- |
| // Phương thức vẽ khung GanttChar không có thời gian đến  public void VeKhung\_NonArrivalTime()  {  Console.Write("\n");  for (int i = 0; i < n \* 16; i++)  {  Console.Write("-");  }  Console.Write("-");  Console.Write("\n");  }  // Phương thức vẽ GanttChar không có thời gian đến  public void GanttChar\_NonArrivalTime()  {  Console.WriteLine("\n\tGANTT CHART");  VeKhung\_NonArrivalTime();  // In ra tên của các tiến trình  for (int i = 0; i < n; i++)  {  Console.Write("|\tP{0}\t", \_dsTT[i].ID);  }  Console.Write("|");  VeKhung\_NonArrivalTime();  // In ra các mốc thời gian hệ thống  Console.Write("0\t\t");  for (int i = 0; i < n; i++)  {  Console.Write("{0}\t\t", \_dsTT[i].TurnaroundTime); // thời gian hoàn  thành của tiến trình  }  VeKhung\_NonArrivalTime();  } |

***Điều phối tiến trình theo giải thuật FCFS có thời gian đến***

|  |
| --- |
| // Điều phối tiến trình theo giải thuật FCFS có thời gian đến  public void DieuPhoiFCFS\_ArrivalTime()  {  \_sumWaitTime = 0; \_sumTurnaroundTime = 0;  Console.WriteLine("\n\t\t\t ----- GIAI THUAT FIRST COME FIRST SERVED  (Arrival Time) ----- ");  SapXepTT\_ArrivalTime(); // Sắp xếp tiến trình theo thời gian đến  // Tính thời gian kết thúc và thời gian chờ của tiến trình đầu tiên  \_dsTT[0].FinishTime = \_dsTT[0].ArrivalTime + \_dsTT[0].BurstTime;  \_dsTT[0].WaitTime = 0;  // Tính thời gian kết thúc và thời gian chờ của các tiến trình còn lại  for (int i = 1; i < n; i++)  {  /\* Nếu thời gian đến của tiến trình này lớn hơn thời gian kết thúc của  tiến trình trước đó thì thời gian kết thúc của tiến trình này = thời  gian đến của tiến trình + thời gian xử lý của tiến trình \*/  if (\_dsTT[i].ArrivalTime > \_dsTT[i - 1].FinishTime)  {  \_dsTT[i].FinishTime = \_dsTT[i].ArrivalTime + \_dsTT[i].BurstTime;  }  /\* Ngược lại thì thời gian kết thúc của tiến trình = thời gian kết thúc  của tiến trình trước + thời gian xử lý của tiến trình \*/  else  {  \_dsTT[i].FinishTime = \_dsTT[i - 1].FinishTime + \_dsTT[i].BurstTime;  }  /\* Tính thời gian chờ của tiến trình = thời gian kết thúc của tiến  trình trước - thời gian đến của tiến trình \*/  \_dsTT[i].WaitTime = \_dsTT[i - 1].FinishTime - \_dsTT[i].ArrivalTime;  /\* Phép trừ có thể có số âm do thời gian đến của tiến trình lớn hơn  thời gian kết thúc của tiến trình trước nên đặt lại thời gian chờ =0 \*/  if (\_dsTT[i].WaitTime < 0)  {  \_dsTT[i].WaitTime = 0;  }  }  /\* Tính thời gian hoàn thành của các tiến trình (= thời gian kết thúc –  thời gian đến của tiến trình đó) \*/  for (int i = 0; i < n; i++)  {  \_dsTT[i].TurnaroundTime = \_dsTT[i].FinishTime - \_dsTT[i].ArrivalTime;  // Tính tổng thời gian chờ của các tiến trình  \_sumWaitTime += \_dsTT[i].WaitTime;  // Tính tổng thời gian hoàn thành của các tiến trình  \_sumTurnaroundTime += \_dsTT[i].TurnaroundTime;  }  } |

***Điều phối tiến trình theo theo SJF NonPreemptive***

|  |
| --- |
| // Điều phối tiến trình theo theo SJF NonPreemptive  public void DieuPhoiSJF\_NonPreemptive()  {  // danh sách để kiểm tra tiến trình đã hoàn thành hay chưa  int[] f = new int[n];  int st = 0, tot = 0; // st: thời gian hệ thống  \_sumWaitTime = 0; \_sumTurnaroundTime = 0;  Console.Write("\n\t\t\t ----- GIAI THUAT SHORTEST FIRST JOB (NonPreemptive)  ----- \n");  for (int i = 0; i < n; i++)  {  f[i] = 0;  \_dsTT[i].FinishTime = 0;  \_dsTT[i].TurnaroundTime = 0;  \_dsTT[i].WaitTime = 0;  }  while (tot != n) // trong khi tot còn khác số tiến trình thì thực hiện  {  int c = n, min = int.MaxValue;  for (int i = 0; i < n; i++)  {  /\* nếu thời gian xuất hiện của tiến trình thứ i <= thời điểm hệ  thống và được đánh dấu f = 0 và thời gian thực hiện bt<min, Tiến  trình đó sẽ được xử lý \*/  if ((\_dsTT[i].ArrivalTime <= st) && (f[i] == 0) &&  (\_dsTT[i].BurstTime < min))  {  min = \_dsTT[i].BurstTime;  c = i;  }  }  /\* Nếu c==n nghĩa là không có tiến trình nào thỏa điều kiện trên nên  tăng thời gian hệ thống \*/  if (c == n)  {  st++;  }  else  {  // thời gian kết thúc = thời gian hệ thống + thời gian xử lý  \_dsTT[c].FinishTime = st + \_dsTT[c].BurstTime;  st += \_dsTT[c].BurstTime;  // thời gian hoàn thành = thời gian kết thúc - thời gian đến  \_dsTT[c].TurnaroundTime = \_dsTT[c].FinishTime –  \_dsTT[c].ArrivalTime;  // thời gian chờ = thời gian hoàn thành - thời gian xử lý  \_dsTT[c].WaitTime = \_dsTT[c].TurnaroundTime - \_dsTT[c].BurstTime;  f[c] = 1; // đánh đấu là tiến trình đã hoàn thành  tot++; // tăng tot  }  }  SapXepTT\_FinishTime();  // Tính toán thời tổng thời gian chờ và tổng thời gian hoàn thành  for (int i = 0; i < n; i++)  {  \_sumWaitTime += \_dsTT[i].WaitTime;  \_sumTurnaroundTime += \_dsTT[i].TurnaroundTime;  }  } |

***Phương thức in ra kết quả điều phối theo FCFS và SJF có thời gian đến***

|  |
| --- |
| // Phương thức in ra kết quả điều phối theo FCFS và SJF có thời gian đến  public void Xuat\_ArrivalTime()  {  // Vẽ sơ đồ gantt  GanttChar\_ArrivalTime();  // In ra thông tin của các tiến trình cũng như thứ tự vào của chúng (được  sắp xếp theo thời gian đến)  Console.WriteLine("\nBang thong tin cua cac tien trinh\n");  Console.WriteLine("| Tien trinh | Thoi gian đen | Thoi gian xu ly | Thoi  gian cho | Thoi gian hoan thanh | Thoi gian ket thuc |");  for (int i = 0; i < n; i++)  {  Console.WriteLine("|\tP{0}\t|\t{1}\t|\t {2}\t |\t {3}\t |\t {4}  \t |\t {5}\t |", \_dsTT[i].ID, \_dsTT[i].ArrivalTime,  \_dsTT[i].BurstTime, \_dsTT[i].WaitTime, \_dsTT[i].TurnaroundTime,  \_dsTT[i].FinishTime);  }  // In ra thời gian chờ trung bình và thời gian hoàn thành trung bình  Console.WriteLine("\n Thoi gian cho trung binh: {0:0.0.0} ms", \_sumWaitTime  \* 1.0 / n);  Console.WriteLine(" Thoi gian hoan thanh trung binh: {0:0.0.0} ms",  \_sumTurnaroundTime \* 1.0 / n);  Console.WriteLine("--------------------------------------------------------  ------------------------------");  } |

***Phương thức vẽ khung GanttChar có thời gian đến***

|  |
| --- |
| // Phương thức vẽ khung GanttChar có thời gian đến  public void VeKhung\_ArrivalTime()  {  count = 0;  Console.Write("\n");  // Nếu thời gian đến của tiến trình > 0 thì tăng biến đếm lên 1 đơn vị  if (\_dsTT[0].ArrivalTime > 0) count++;  for (int i = 1; i < n; i++)  {  // Nếu thời gian đến của tiến trình > thời gian kết thúc của tiến trình  trước thì tăng biến đếm lên 1 đơn vị  if (\_dsTT[i].ArrivalTime > \_dsTT[i - 1].FinishTime) {  count++;  }  }    for (int i = 0; i < (n + count) \* 16; i++)  {  Console.Write("-");  }  Console.Write("-");  Console.Write("\n");  } |

***Phương thức vẽ GanttChar có thời gian đến***

|  |
| --- |
| // Phương thức vẽ GanttChar có thời gian đến  public void GanttChar\_ArrivalTime()  {  Console.WriteLine("\n\tGANTT CHART");  VeKhung\_ArrivalTime();  // In ra tên của các tiến trình  // Nếu thời gian đến của tiến trình đầu tiên > 0 thì thêm 1 ô phía trước  if (\_dsTT[0].ArrivalTime > 0)  {  Console.Write("|\t\t|\tP{0}\t", \_dsTT[0].ID);  }  else Console.Write("|\tP{0}\t", \_dsTT[0].ID);  for (int i = 1; i < n; i++)  {  if (\_dsTT[i].ArrivalTime <= \_dsTT[i - 1].FinishTime) {  Console.Write("|\tP{0}\t", \_dsTT[i].ID);  }  // Nếu thời gian đến của tiến trình > thời gian kết thúc của tiến trình  trước thì thêm 1 ô phía trước  else {  Console.Write("|\t\t|\tP{0}\t", \_dsTT[i].ID);  }  }  Console.Write("|");  VeKhung\_ArrivalTime();  // In ra các mốc thời gian hệ thống  if (\_dsTT[0].ArrivalTime > 0) {  Console.Write("0\t\t{0}\t\t{1}\t\t", \_dsTT[0].ArrivalTime,  \_dsTT[0].FinishTime);  }  else {  Console.Write("{0}\t\t{1}\t\t", \_dsTT[0].ArrivalTime,  \_dsTT[0].FinishTime);  }  for (int i = 1; i < n; i++)  {    if (\_dsTT[i].ArrivalTime <= \_dsTT[i - 1].FinishTime) {  Console.Write("{0}\t\t", \_dsTT[i].FinishTime);  }  // Nếu thời gian đến của tiến trình > thời gian kết thúc của tiến  trình trước thì in ra thời gian đến và thời gian kết thúc của tiến  trình đó  else {  Console.Write("{0}\t\t{1}\t\t", \_dsTT[i].ArrivalTime,  \_dsTT[i].FinishTime);  }  }  VeKhung\_ArrivalTime();  } |

***Phương thức sắp xếp tăng dần các tiến trình theo thời gian đến***

|  |
| --- |
| // Phương thức sắp xếp tăng dần các tiến trình theo thời gian đến  public void SapXepTT\_ArrivalTime()  {  for (int i = 0; i < n - 1; i++)  {  for (int j = i + 1; j < n; j++)  {  if (\_dsTT[i].ArrivalTime > \_dsTT[j].ArrivalTime)  {  TienTrinh temp = new TienTrinh();  temp = \_dsTT[i];  \_dsTT[i] = \_dsTT[j];  \_dsTT[j] = temp;  }  }  }  } |

***Phương thức sắp xếp tăng dần các tiến trình theo thời gian xử lý***

|  |
| --- |
| // Phương thức sắp xếp tăng dần các tiến trình theo thời gian xử lý  public void SapXepTT\_BurstTime()  {  for (int i = 0; i < n - 1; i++)  {  for (int j = i + 1; j < n; j++)  {  if (\_dsTT[i].BurstTime > \_dsTT[j].BurstTime)  {  TienTrinh temp = new TienTrinh();  temp = \_dsTT[i];  \_dsTT[i] = \_dsTT[j];  \_dsTT[j] = temp;  }  }  }  } |

***Phương thức sắp xếp tăng dần các tiến trình theo thời gian hoàn thành***

|  |
| --- |
| // Phương thức sắp xếp tăng dần các tiến trình theo thời gian hoàn thành  public void SapXepTT\_FinishTime()  {  for (int i = 0; i < n - 1; i++)  {  for (int j = i + 1; j < n; j++)  {  if (\_dsTT[i].FinishTime > \_dsTT[j].FinishTime)  {  TienTrinh temp = new TienTrinh();  temp = \_dsTT[i];  \_dsTT[i] = \_dsTT[j];  \_dsTT[j] = temp;  }  }  }  } |

***Phương thức sắp xếp tăng dần các tiến trình theo ID***

|  |
| --- |
| // Phương thức sắp xếp tăng dần các tiến trình theo ID  public void SapXepTT\_ID()  {  for (int i = 0; i < n - 1; i++)  {  for (int j = i + 1; j < n; j++)  {  if (\_dsTT[i].ID > \_dsTT[j].ID)  {  TienTrinh temp = new TienTrinh();  temp = \_dsTT[i];  \_dsTT[i] = \_dsTT[j];  \_dsTT[j] = temp;  }  }  }  } |

### Class Program

|  |
| --- |
| class Program  {  static void Main()  {  // tạo 1 thực thể của lớp DSTienTrinh  DSTienTrinh dstt = new DSTienTrinh();    // gọi phương thức nhập Burst time cho danh sách tiến trình  dstt.NhapBurstTime\_DSTT();  bool kt = false; // kiểm tra xem đã nhập Arrival Time vào danh sách  tiến trình chưa  // Menu chính để người dùng lựa chọn giải thuật  MenuOption:  Console.WriteLine("\n\tMoi ban chon Giai Thuat Mo Phong :");  Console.WriteLine("\t-[1]. Giai Thuat FCFS");  Console.WriteLine("\t-[2]. Giai Thuat SJF");  Console.WriteLine("\t-[Other]. Thoat chuong trinh");  Console.Write("\t ==> Ban chon: ");  char checkOption = char.Parse(Console.ReadLine()); // lưa lựa chọn của  người dùng vào biến checkOption  switch (checkOption)  {  // Người dùng chọn 1 thì hiển thị menu FCFS  case '1':  // Menu FCFS để người dùng lựa chọn loại FCFS  FCFSMenu:  Console.WriteLine("\n\tMoi ban chon loai FCFS: ");  Console.WriteLine("\t-[1]. FCFS Non Arrival Time");  Console.WriteLine("\t-[2]. FCFS Arrival Time");  Console.Write("\t ==> Ban chon: ");  char FCFSOption = char.Parse(Console.ReadLine());  // Nếu người dùng chọn 1 thì điều phối theo  FCFS\_NonArrivalTime và xuất tương ứng  if (FCFSOption == '1')  {  dstt.DieuPhoiFCFS\_NonArrivalTime();  dstt.Xuat\_NonArrivalTime();  }  /\* Nếu người dùng chọn 2 thì kiểm tra đã nhập thời gian đến  chưa, nếu chưa thì sắp xếp lại tiến trình theo ID và nhập thời  gian đến sau đó điều phối theo FCFS\_ArrivalTime và xuất tương  ứng \*/  else if (FCFSOption == '2')  {  if (kt == false)  {  dstt.SapXepTT\_ID();  dstt.NhapArrivalTimeDSTT();  kt = true;  }  dstt.DieuPhoiFCFS\_ArrivalTime();  dstt.Xuat\_ArrivalTime();  }  else { Console.WriteLine("Ban chon khong co trong menu"); goto  FCFSMenu; }  /\* Nếu người dùng chọn 1 hoặc 2 rồi thì hỏi người dùng có muốn  mô phỏng loại khác không hay về Menu chính \*/  if (FCFSOption == '1' || FCFSOption == '2')  {  Console.WriteLine(" Ban Co Muon Mo Phong FCFS Khac? \n(  Nhan 'Y' de dong y hoac [Phim Bat Ki] khac de quay tro ve  Menu chuong trinh roi [ENTER])");  char checkOtherOption;  checkOtherOption = char.Parse(Console.ReadLine());    // Nếu người dùng nhập y hoặc Y thì quay về FCFSMenu  if (checkOtherOption == 'y' || checkOtherOption == 'Y') {  goto FCFSMenu;  }  // Nếu không thì quay về Menu chính  else {  goto MenuOption;  }  }  break;  // Người dùng chọn 2 thì hiển thị menu SJF  case '2':  // Menu SJF để người dùng lựa chọn loại SJF  SJFMenu:  Console.WriteLine("\n\tMoi ban chon loai SJF: ");  Console.WriteLine("\t-[1]. SJF Non Arrival Time");  Console.WriteLine("\t-[2]. SJF NonPreemptive (Arrival Time) ");  Console.Write("\t ==> Ban chon: ");  char SJFOption = char.Parse(Console.ReadLine());  // Nếu người dùng chọn 1 thì điều phối theo  SJF\_NonArrivalTime và xuất tương ứng  if (SJFOption == '1')  {  dstt.DieuPhoiSJF\_NonArrivalTime();  dstt.Xuat\_NonArrivalTime();  }  /\* Nếu người dùng chọn 2 thì kiểm tra đã nhập thời gian đến  chưa, nếu chưa thì sắp xếp lại tiến trình theo ID và nhập thời  gian đến sau đó điều phối theo SJF\_NonPreemptive và xuất tương  ứng \*/  else if (SJFOption == '2')  {  if (kt == false)  {  dstt.SapXepTT\_ID();  dstt.NhapArrivalTimeDSTT();  kt = true;  }  dstt.DieuPhoiSJF\_NonPreemptive();  dstt.Xuat\_ArrivalTime();  }  else { Console.WriteLine("Ban chon khong co trong menu"); goto  SJFMenu; }  /\* Nếu người dùng chọn 1 hoặc 2 rồi thì hỏi người dùng có muốn  mô phỏng loại khác không hay về Menu chính \*/  if (SJFOption == '1' || SJFOption == '2')  {  Console.WriteLine(" Ban Co Muon Mo Phong SJF Khac? \n( Nhan  'Y' de dong y hoac [Phim Bat Ki] khac de quay tro ve Menu  chuong trinh roi [ENTER])");  char checkOtherOption;  checkOtherOption = char.Parse(Console.ReadLine());  // Nếu người dùng nhập y hoặc Y thì quay về SJFMenu  if (checkOtherOption == 'y' || checkOtherOption == 'Y') {  goto SJFMenu;  }  // Nếu không thì quay về Menu chính  else {  goto MenuOption;  }  }  break;  // Người dùng chọn khác thì thoát chương trình  default:  Console.WriteLine("\nBan da thoat chuong trinh!");  break;  }  }  } // End of class Program |

## Đầu vào và đầu ra của chương trình

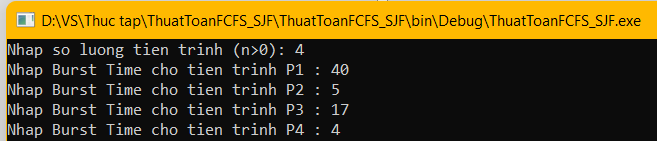
* Đầu vào:
* Số lượng tiến trình cần xử lý
* Thời gian xử lý cho từng tiến trình
* Nếu chọn có thời gian đến thì nhập thời gian đến cho từng tiến trình
* Đầu ra:
* Hiển thị sơ đồ Gantt
* Bảng thông tin các tiến trình
* Thời gian chờ trung bình và thời gian hoàn thành trung bình

## Các chức năng của chương trình

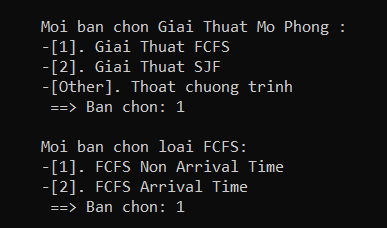
* Chọn giải thuật muốn mô phỏng
* In ra kết quả cho từng giải thuật tương ứng

## Cài đặt chương trình

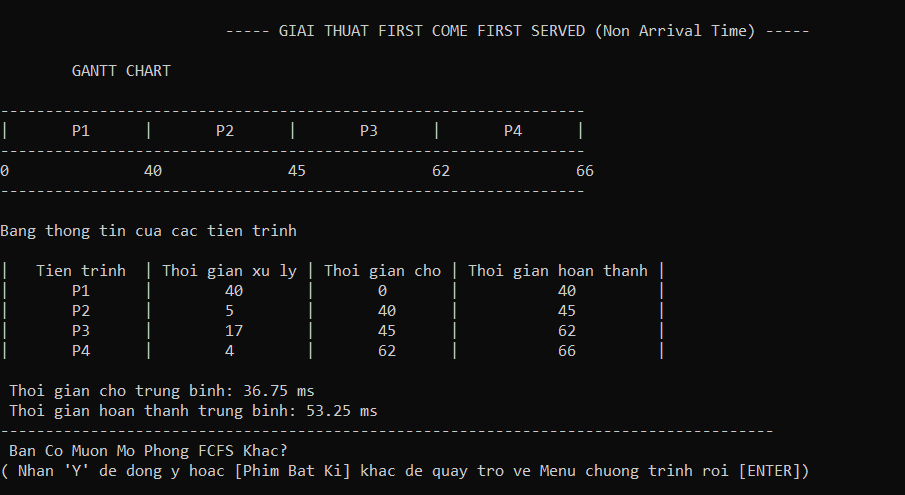
* Đầu vào



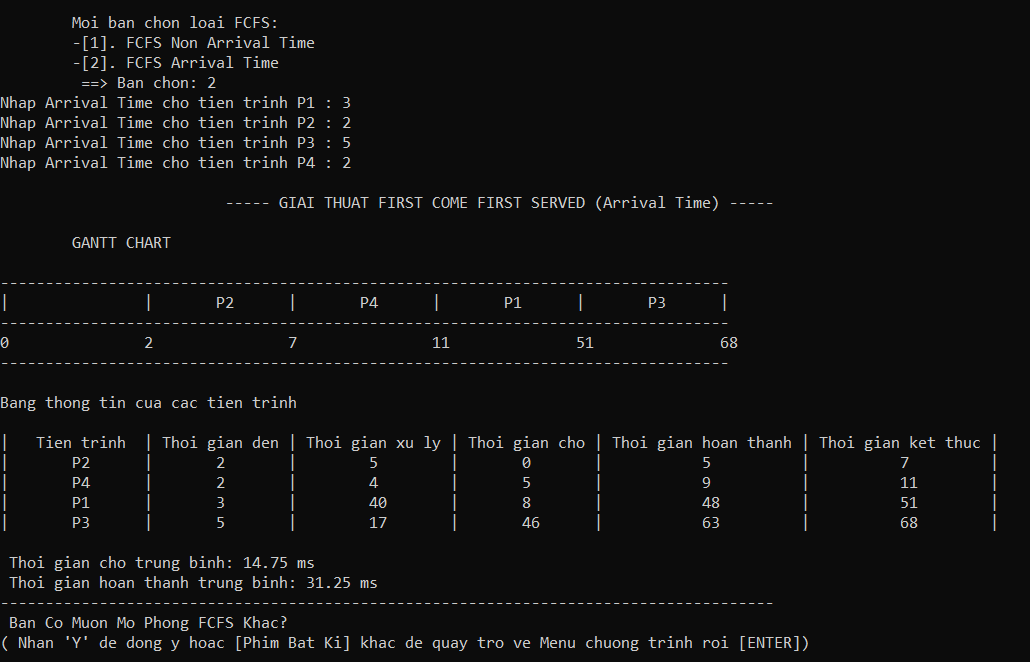
* Lựa chọn giải thuật mô phỏng:



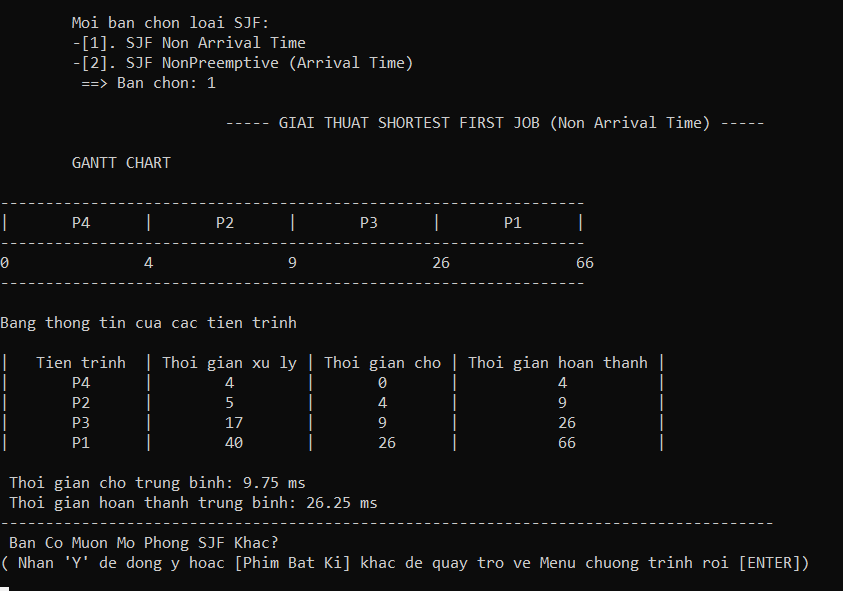
* Chọn 1 để thực hiện giải thuật FCFS. Tiếp theo
* chọn 1 để thực hiện FCFS không có thời gian đến
* chọn 2 để thực hiện FCFS có thời gian đến
* Chọn 2 để thực hiện giải thuật SJF. Tiếp theo
* chọn 1 để thực hiện SJF không có thời gian đến
* chọn 2 để thực hiện SJF\_NonPreemtive (có thời gian đến)
* FCFS Non Arrival Time



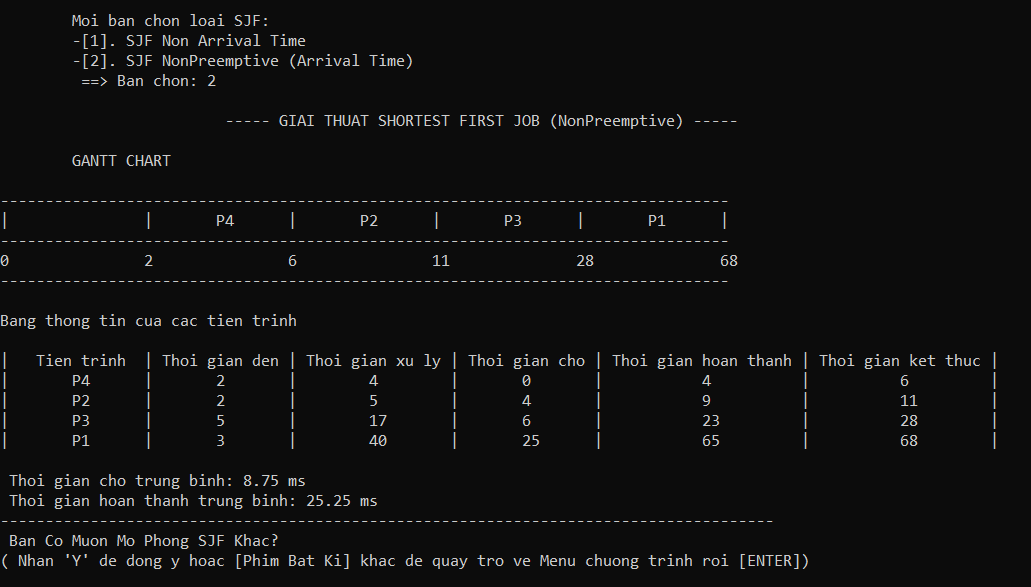
* FCFS Arrival Time



* SJF Non Arrival Time



* SJF Arrival Time



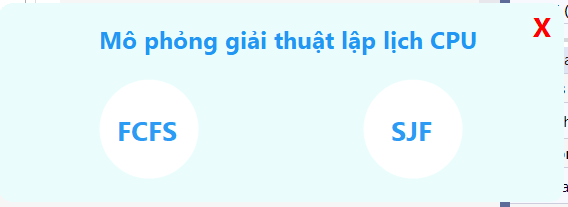
# XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG BẰNG GIAO DIỆN

## Môi trường phát triển

* Ứng dụng được phát triển trên Visual Studio với project Windows Forms App (.NET Framework).
* Cài đặt thêm packages Guna.UI2.WinForms

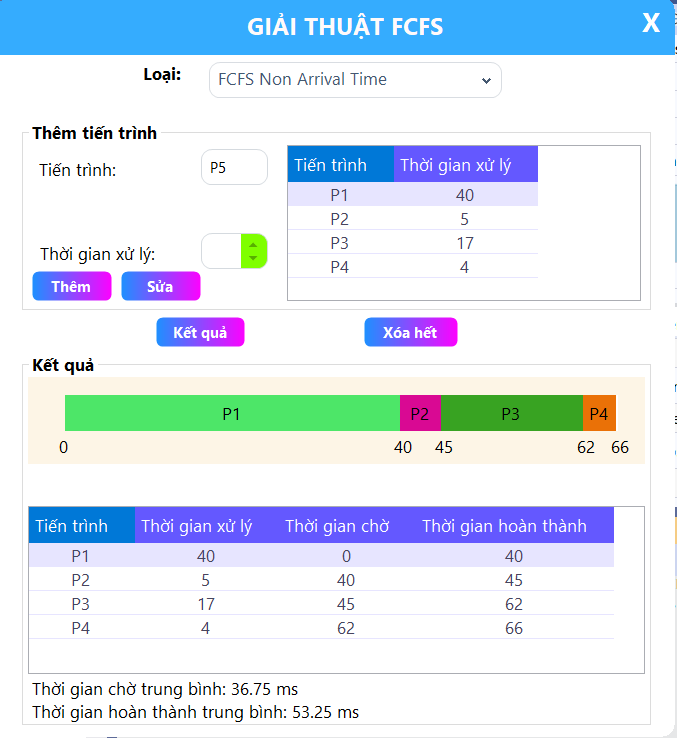
## Giao diện của chương trình

### Form menu chính

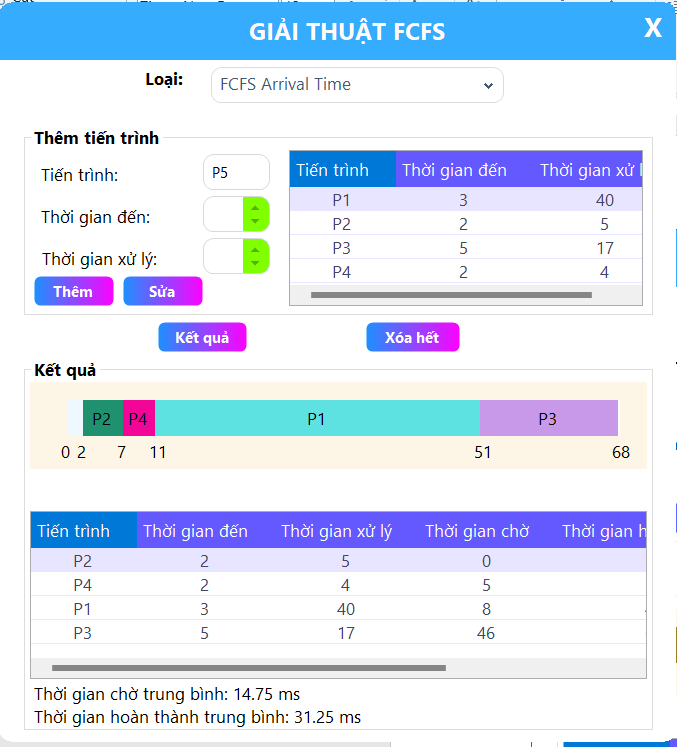


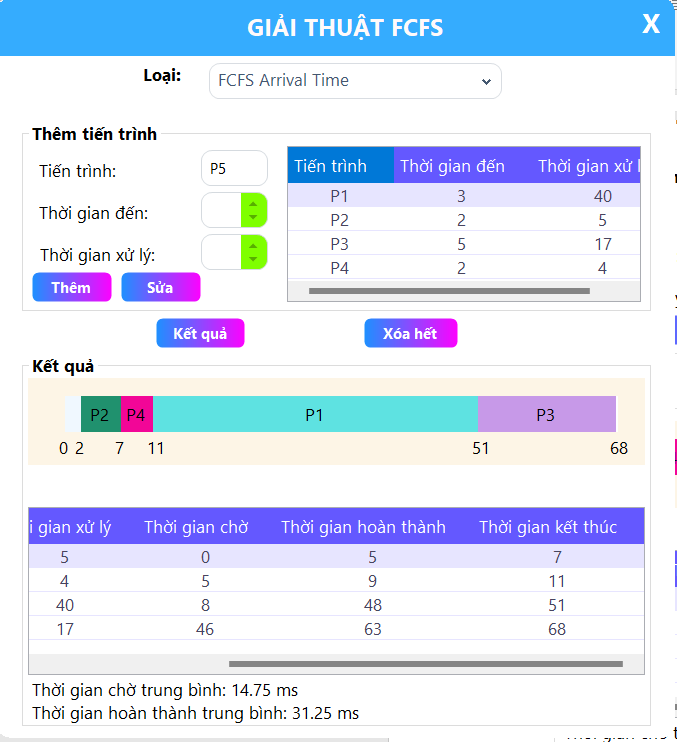
### Form FCFS

#### FCFS Non Arrival Time



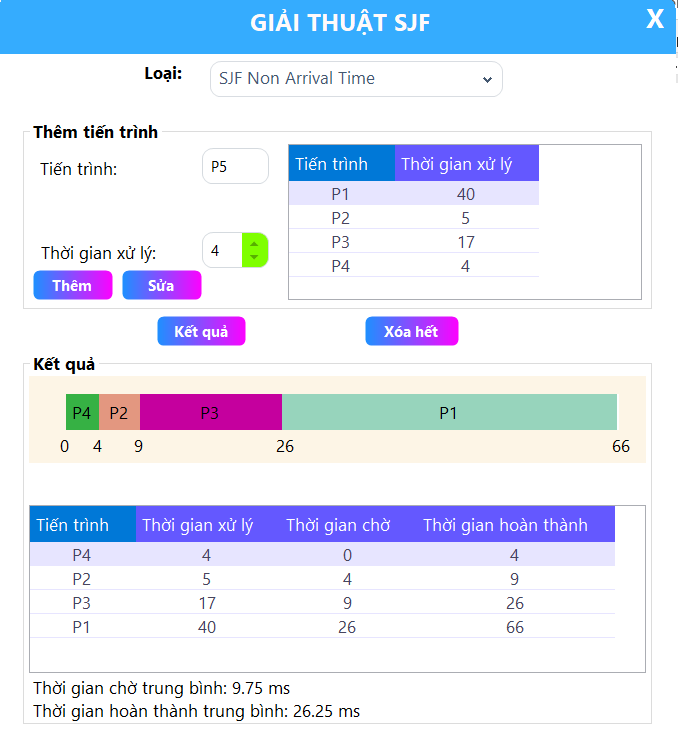
#### FCFS Arrival Time



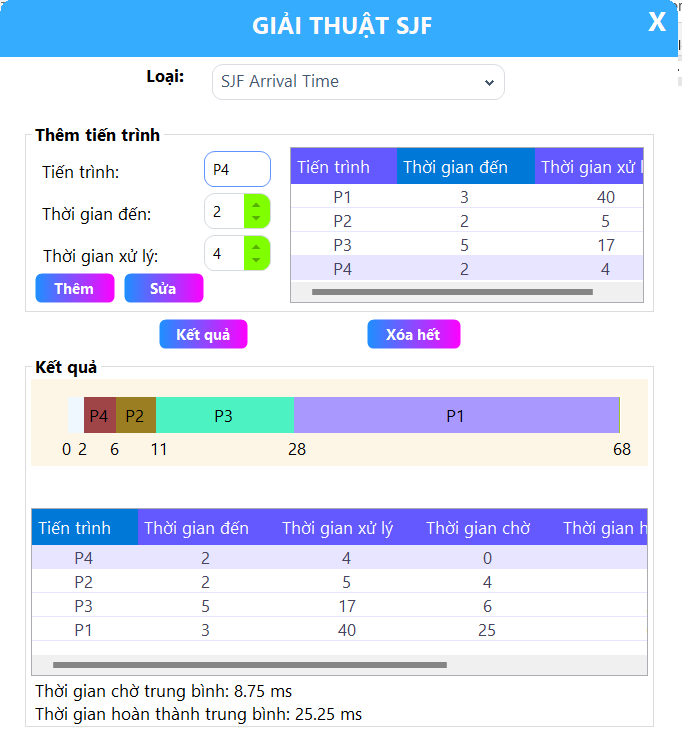


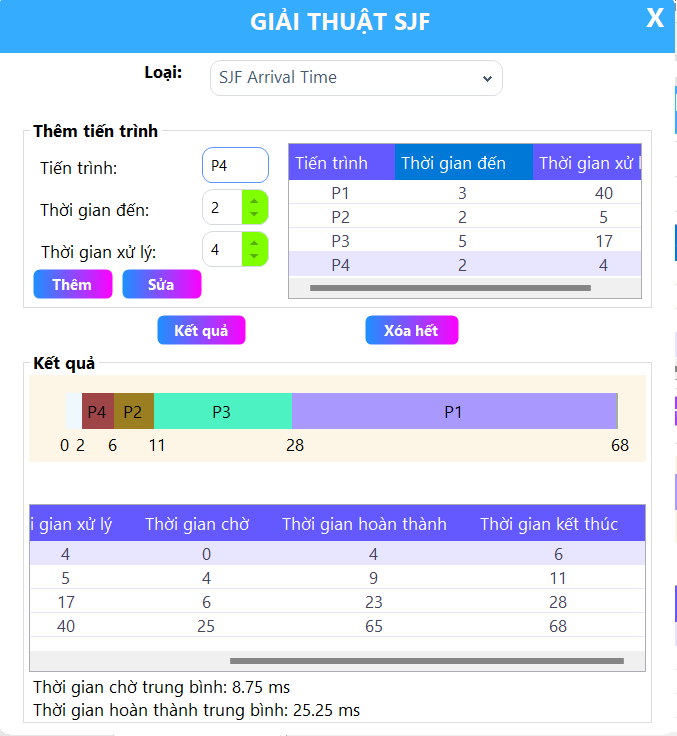
### Form SJF

#### SJF Non Arrival Time



#### SJF NonPreemptive (Arrival Time)





# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## KẾT QUẢ ĐẠT ĐƯỢC

* Trình bày lý thuyết thuật toán FCFS và SJF.
* Cài đặt các bước thực hiện thuật toán FCFS và SJF bằng ngôn ngữ lập trình C#.
* Xây dựng được chương trình mô phỏng đồ họa với Windows Forms.

## ƯU ĐIỂM

* Chương trình dễ sử dụng, giao diện thân thiện với người dùng
* Tốc độ xử lý nhanh, độ chính xác cao.

## HẠN CHẾ

* + Code chương trình còn dài, chưa tối ưu.

## HƯỚNG PHÁT TRIỂN

Mô phỏng thêm các giải thuật lập lịch CPU khác như Priority, Round Robin, SRTF (Shortest Remaining Time First), xổ số (Lottery),…

## KẾT LUẬN

* Đã trình bày được lý thuyết thuật toán FCFS và SJF.
* Xây dựng được chương trình mô phỏng các bước thực hiện thuật toán FCFS và SJF sử dụng giao diện text và đồ họa .
* Tương lai sẽ mô phỏng thêm các giải thuật lập lịch CPU khác.

# TÀI LIỆU THAM KHẢO

Slide bài giảng “Hệ điều hành” – Lương Trần Hy Hiến (trường đại học Công Nghệ Tp.HCM)

Slide bài giảng “Hệ điều hành” – Nguyễn Khắc Cường (trường đại học Nha Trang)

Slide bài giảng “Nguyên lý hệ điều hành” – Nguyễn Thị Quỳnh Hoa (trường đại học Sư Phạm Hà Nội)

<https://sinhvientot.net/chien-luoc-dieu-phoi-cpu-first-come-first-server/>

<https://sinhvientot.net/giai-thuat-dieu-phoi-shortest-job-first-scheduling-sjf/>

<https://viblo.asia/p/he-dieu-hanh-may-tinh-hoat-dong-nhu-the-nao-phan-1-aWj534BbK6m>

<https://huongtlu.wordpress.com/2020/07/02/thuat-toan-fcfs-sjf-srt-vi-du-chi-tiet/>

<https://viblo.asia/p/cac-tien-trinh-trong-he-dieu-hanh-may-tinh-bWrZnQerKxw>

<https://viblo.asia/p/tien-trinh-trong-he-dieu-hanh-phan-2-yMnKMqOzK7P>

<https://viblo.asia/p/tien-trinh-trong-he-dieu-hanh-phan-3-3Q75wg6Q5Wb>

<https://voer.edu.vn/c/dieu-phoi-tien-trinh/a039fa79/ba6e1a94>

Slide bài giảng “Lập trình hướng đối trượng” – Nguyễn Đình Hưng (trường đại học Nha Trang)

<https://vietnix.vn/c-sharp-la-gi/>

<https://www.dtechvn.com/c-la-gi-tong-quan-ve-c.html>