# Sự khác biệt giữa Semaphore và Mutex

Đồng bộ hóa tiến trình đóng vai trò quan trọng trong việc duy trì tính nhất quán của dữ liệu được chia sẻ. Cả hai giải pháp phần mềm và phần cứng đều được sử dụng để xử lý vấn đề này. Tuy nhiên các giải pháp phần cứng thường khó thực hiện. Trong bài viết này, chúng tôi sẽ thảo luận về hai giải pháp dựa trên phần mềm để xử lý vấn đề quan trọng là Semaphore và Mutex.

Sự khác biệt cơ bản giữa semaphore và mutex là: semaphore là một cơ chế báo hiệu, tức là các tiến trình thực hiện thao tác wait () và signal () để cho biết liệu chúng có được sử dụng hoặc giải phóng tài nguyên hay không, trong khi Mutex là cơ chế khóa, tiến trình phải thu được khóa đối tượng mutex nếu nó muốn có được tài nguyên. Hãy xem biểu đồ so sánh dưới đây.

### Biểu đồ so sánh

|  |  |
| --- | --- |
| **Semaphore** | **Mutex** |
| Semaphore là một cơ chế báo hiệu. | Mutex là một cơ chế khóa. |
| Semaphore là một biến số nguyên. | Mutex là một đối tượng. |
| Semaphore có thể được phân loại thành Counting semaphore và Binary semaphore. | Mutex không được phân loại thêm. |
| Giá trị semaphore được sửa đổi bằng cách sử dụng các thao tác wait () và signal (). | Đối tượng Mutex bị khóa hoặc mở khóa bởi tiến trình yêu cầu hoặc giải phóng tài nguyên. |
| Nếu tất cả các tài nguyên đang được sử dụng, tiến trình yêu cầu tài nguyên thực hiện thao tác wait () và tự chặn cho đến khi số lượng semaphore trở nên lớn hơn một. | Nếu một đối tượng mutex đã bị khóa, tiến trình yêu cầu tài nguyên chờ và được hệ thống xếp hàng cho đến khi khóa được giải phóng. |

### Định nghĩa của Semaphore

Semaphore là một công cụ đồng bộ hóa tiến trình. Semaphore thường là **biến số nguyên** **S** được khởi tạo cho số lượng tài nguyên có trong hệ thống và giá trị của semaphore chỉ có thể được **sửa đổi** bằng hai hàm **wait ()** và **signal ()** ngoài khởi tạo.

Hoạt động wait () và signal () sửa đổi giá trị của semaphore **không thể tách rời** . Nó có nghĩa là khi một tiến trình đang sửa đổi giá trị của semaphore, không có tiến trình nào khác có thể đồng thời sửa đổi giá trị của semaphore. Semaphore được phân biệt bởi hệ điều hành trong hai loại **Đếm semaphores** và **Baph semaphore** .

Trong **Đếm Semaphore**, giá trị semaphore S được khởi tạo cho **số lượng tài nguyên** s có trong hệ thống. Bất cứ khi nào một tiến trình muốn truy cập vào tài nguyên, nó thực hiện thao tác wait **()** trên semaphore và **giảm** giá trị của semaphore bởi một. Khi nó giải phóng tài nguyên, nó thực hiện thao tác **signal ()** trên semaphore và **tăng** giá trị của semaphore lên một. Khi số lượng semaphore về 0, điều đó có nghĩa là tất cả các tài nguyên bị chiếm bởi các quy trình. Nếu một tiến trình cần sử dụng tài nguyên khi số semaphore bằng 0, thì nó thực thi wait () và **bị chặn** cho đến khi giá trị của semaphore trở nên lớn hơn 0.

Trong **semaphore nhị phân**, giá trị của semaphore nằm trong khoảng từ **0** đến **1** . Nó tương tự như khóa mutex, nhưng mutex là một cơ chế khóa trong khi đó, semaphore là một cơ chế báo hiệu. Trong semaphore nhị phân, nếu một tiến trình muốn truy cập vào tài nguyên, nó thực hiện thao tác wait **()** trên semaphore và giảm giá trị của semaphore từ 1 xuống 0. Khi nó giải phóng tài nguyên, nó sẽ thực hiện thao tác **signal ()** trên semaphore và tăng dần giá trị của nó là 1. Nếu giá trị của semaphore là 0 và một tiến trình muốn truy cập vào tài nguyên thì nó thực hiện thao tác **wait ()** và tự chặn cho đến khi quy trình hiện tại sử dụng tài nguyên giải phóng tài nguyên.

### Định nghĩa của Mutex

**Đối tượng loại trừ lẫn nhau** được gọi ngắn gọn là Mutex. Từ thuật ngữ loại trừ lẫn nhau, chúng ta có thể hiểu rằng chỉ một tiến trình tại một thời điểm có thể truy cập vào tài nguyên đã cho. Đối tượng mutex cho phép nhiều luồng chương trình sử dụng cùng một tài nguyên nhưng mỗi lần một luồng không đồng thời.

Khi một chương trình khởi động, nó yêu cầu hệ thống tạo một đối tượng mutex cho một tài nguyên nhất định. Hệ thống tạo đối tượng mutex với một tên hoặc ID duy nhất. Bất cứ khi nào luồng chương trình muốn sử dụng tài nguyên mà nó chiếm khóa trên đối tượng mutex, sử dụng tài nguyên và sau khi sử dụng, nó sẽ giải phóng khóa trên đối tượng mutex. Sau đó, tiến trình tiếp theo được phép có được khóa trên đối tượng mutex.

Trong khi đó, một tiến trình đã có được khóa trên đối tượng mutex, không có luồng / tiến trình nào khác có thể truy cập vào tài nguyên đó. Nếu đối tượng mutex đã bị khóa, tiến trình mong muốn có được khóa trên đối tượng mutex phải chờ và được hệ thống xếp hàng chờ đợi cho đến khi đối tượng mutex được mở khóa.

## Sự khác biệt chính giữa Semaphore và Mutex

1. Semaphore là một cơ chế **báo hiệu** khi hoạt động wait () và signal () được thực hiện trên biến semaphore cho biết liệu một tiến trình có thu được tài nguyên hay giải phóng tài nguyên hay không. Mặt khác, mutex là một cơ chế **khóa**, vì để có được một tài nguyên, một tiến trình cần phải khóa đối tượng mutex và trong khi phát hành một quy trình tài nguyên phải mở khóa đối tượng mutex.
2. Semaphore thường là một biến **số nguyên** trong khi đó, mutex là một **đối tượng** .
3. Semaphore cho phép nhiều luồng chương trình truy cập vào **tài nguyên hữu hạn** . Mặt khác, Mutex cho phép nhiều luồng chương trình truy cập vào một **tài nguyên được chia sẻ** nhưng mỗi lần một **tài nguyên** .
4. Giá trị biến semaphore có thể được sửa đổi bởi **bất kỳ** tiến trình nào có được hoặc giải phóng tài nguyên bằng cách thực hiện thao tác wait () và signal (). Mặt khác, khóa thu được trên đối tượng mutex chỉ có thể được giải phóng bằng tiến trình đã thu được khóa trên đối tượng mutex.
5. Semaphore có hai loại **đếm semaphore và semaphore nhị phân** khá giống với mutex.
6. Giá trị biến semaphore được sửa đổi bằng thao tác **wait ()** và **signal ()** ngoài khởi tạo. Tuy nhiên, đối tượng tắt tiếng bị khóa hoặc mở khóa bằng tiến trình lấy hoặc giải phóng tài nguyên.
7. Nếu tất cả các tài nguyên được thu thập bởi quy trình và không có tài nguyên nào là miễn phí thì quy trình mong muốn có được tài nguyên thực hiện thao tác wait () trên biến semaphore và tự **chặn** cho đến khi số lượng semaphore lớn hơn 0. Nhưng nếu một đối tượng mutex đã bị khóa sau đó tiến trình mong muốn có được tài nguyên **chờ đợi** và được hệ thống **xếp hàng** cho đến khi tài nguyên được giải phóng và đối tượng mutex được mở khóa.

### Phần kết luận:

Semaphore là một lựa chọn tốt hơn trong trường hợp có nhiều phiên bản tài nguyên có sẵn. Trong trường hợp mutex tài nguyên chia sẻ duy nhất là một lựa chọn tốt hơn.