Slide 2:

Một process (tiến trình) trong hệ điều hành có thể được tiến hành độc lập hoặc giao tiếp với nhau. Tiến trình độc lập là khi tiến trình không ảnh hưởng hoặc bị ảnh hưởng bởi các tiến trình khác trong hệ thống, và không chia sẻ data với bất kì tiến trình nào. Tiến trình hợp tác khi tiến trình đó có thể ảnh hưởng hoặc bị ảnh hưởng bởi các tiến trình khác trong hệ thống, và sự chia sẻ data có diễn ra.

Lợi ích của tiến trình hợp tác là:

+Giúp chia sẻ thông tin giữa các users.

+Giúp tang tốc độ tính toán

+Giúp xây dựng modun.

+Giúp thuận tiện trong chạy nhiều tác vụ cùng một lúc.

Slide 3:

Vấn đề về dữ liệu được chia sẻ:

+ Việc các tiến trình truy cập đồng thời vào dữ liệu được chia sẻ có thể dẫn đến sự không nhất quán của dữ liệu

+ Để duy trì tính nhất quán của dữ liệu đòi hỏi các cơ chế để đảm bảo thực hiện có trật tự các tiến trình hợp tác.

+ Cần có cơ chế để các tiến trình giao tiếp với nhau và đồng bộ hóa các hành động của chúng

Slide 4:

Trong tiến trình hợp tác, việc chia sẻ dữ liệu cho nhau là rất dễ dàng, nhưng từ đó nó cũng phát sinh nhiều vấn đề liên quan. Một trong số vấn đề đó là điều kiện cuộc đua,

Khi 2 tiến trình hoặc nhiều tiến trình muốn truy cập đồng thời vào bộ nhớ chia sẻ và kết quả cuối cùng phụ thuộc vào tiến trình nào chạy chính xác, khi đó được goi là điều kiện cuộc đua.

Giải thích hình ảnh:

Ta có một thư mục bộ đệm với nhiều khe được đánh số, mỗi khe có khả năng chứa 1 tên tệp. Có 2 biến được chia sẻ là biến out và biến in trong đó biến in sẽ trỏ tới vị trí trống tiếp theo trong thư mục. Tại một thời điểm, các khe từ 0 -> 3 là các khe trống, các khe từ 4 - > 6 là các khe đã đầy ( Với tên các tệp xếp hang đợi để in ). Khi đó có 2 tiến trình A và B sẽ quyết đinh tệp nào sẽ được in.

Quy trình A đọc biến in và lưu trữ giá trị trong một biến cục bộ gọi là vị trí trống tiếp theo.Sau đó hiện tượng ngắt xảy ra, CPU quyết định tiến trình A sẽ dừng lại và chuyển sang tiến trình B. Tiến trình B cũng đọc biến in và cũng nhận được 7, nó cũng lưu trữ tại vị trí giống tiến trình A. Tại đây cả 2 tiến trình đều nghĩ khe khả dụng tiếp theo nằm ở vị trí số 7. Tiến trình B tiếp tục chạy và lưu tên tệp của nó trong khe số 7 và cập nhật biến in thành 8. Cuối cùng, tiến trình A chạy lại, bắt đầu từ vị trí mà nó đã dừng lại. Nó tìm vị khe số 7 và viết tên tệp của nó vào khe 7, xóa tên tệp mà tiến trình B vừa mới đặt ở đó. Như vậy tiến trình B không nhận được bất kì đầu ra nào.

Làm thế nào để ta có thể tránh được điều kiện cuộc đua? Đó là sử dụng giái pháp loại trừ lẫn nhau nghĩa là một số cách để ngăn cản nhiều tiến trình truy cập vào dữ liệu được chia sẻ cùng một lúc, đảm bảo rằng nếu một tiến trình đang sử dụng một biến hoặc tệp được chia sẻ, các tiến trình khác sẽ bị loại trừ khỏi việc thực hiện cùng một việc.

Slide 5:

Một phần thời gian, một tiến trình bận rộn thực hiện các tính toán nội bộ và những thứ khác không dẫn đến điều kiện cuộc đua. Tuy nhiên, đôi khi một tiến trình phải truy cập vào bộ nhớ hoặc tệp được chia sẻ hoặc thực hiện những việc quan trọng khác có thể dẫn đến điều kiện cuộc đua. Phần chương trình mà bộ nhớ dùng chung được truy cập được gọi là vùng trọng yếu hoặc vùng quan trọng. Nếu chúng ta có thể sắp xếp các vấn đề sao cho không có hai tiến trình nào ở trong các khu vực quan trọng của chúng cùng một lúc, chúng ta có thể tránh được các điều kiện cuộc đua.

Mặc dù yêu cầu này tránh được các điều kiện cuộc đua, nhưng nó không đủ để các quá trình song song hợp tác một cách chính xác và hiệu quả bằng cách sử dụng dữ liệu được chia sẻ. Chúng ta cần có bốn điều kiện để có một giải pháp tốt:

1. Không có hai tiến trình nào có thể đồng thời bên trong các vùng quan trọng của chúng.
2. Không có giả định nào được đưa ra về tốc độ hoặc số lượng CPU.
3. Không có tiến trình nào đang chạy bên ngoài vùng quan trọng của nó có thể chặn các tiến trình khác.
4. Không có quá trình nào phải đợi mãi để đi vào vùng quan trọng của nó.

Slide 6:

Ví dụ về loại trừ lẫn nhau bằng cách sử dụng các vùng quan trọng

Giải thích hình ảnh:

Tiến trình A đi vào vùng quan trọng của nó tại thời điểm T1. Một lúc sau, tại thời điểm T2, quá trình B cố gắng đi vào vùng quan trọng của nó nhưng không thành công vì một quá trình khác đã nằm trong vùng quan trọng của nó và do chỉ cho phép một quá trình tại một thời điểm. Do đó, B tạm thời sẽ đợi đến khi A đi ra khỏi vùng quan trọng của nó. Đến thời điểm T3, tiến trình A rời khỏi, khi đó B sẽ được phép đi vào tại lúc đó và cuối cùng B đi ra khỏi vùng quan trọng tại thời điểm T4.

Slide 7: Loại trừ lẫn nhau với chờ đợi bận rộn

Trong phần này, chúng ta sẽ xem xét các đề xuất khác nhau để đạt được loại trừ lẫn nhau, để trong khi một quy trình đang bận cập nhật bộ nhớ dùng chung trong vùng quan trọng của nó, thì không có quy trình nào khác sẽ xâm nhập vào vùng quan trọng của nó và gây ra sự cố.

Đầu tiên là các đề xuất liên quan đến phần mềm:

+Khóa các biến

+Thay thế nghiêm ngặt

+Giai pháp của Peterson

Tiếp theo là các đề xuất liên quan đến phần cứng:

+Vô hiệu hóa các ngắt

+Mã lệnh TSL

Slide 8:

Trước tiên ta đi tìm hiểu các đề xuất phần mềm, thứ nhất là khóa các biến

Giải thích hình ảnh:

+ Ban đầu đặt lock = 0, Khi một quy trình muốn nhập vào vùng quan trọng của nó, trước tiên nó sẽ kiểm tra lock.

+ Nếu lock là 0, tiến trình đặt nó thành 1 và đi vào vùng quan trọng. Nếu lock đã là 1, tiến trình chỉ đợi cho đến khi lock trở thành 0.

+ Do đó, 0 có nghĩa là không có tiến trình nào nằm trong vùng quan trọng của nó và 1 có nghĩa là một số tiến trình nằm trong vùng quan trọng của nó.

Slide 9:

Tiếp theo là đề xuất phần mềm thứ 2 là thay thế nghiêm ngặt

Giải thích hình ảnh:

+Có 1 biến số nguyên ban đầu Turn = 0, theo dõi xem tiến trình nào sẽ đến lượt đi vào vùng quan trọng

+Ban đầu, tiến trình 0 kiểm tra biến Turn, thấy Turn = 0 và đi vào vùng quan trọng của nó, tiến trình 1 cũng kiểm tra biến Turn và nhận thấy Turn = 0, do đó tiến trình 1 sẽ nằm trong một vòng lặp chặt chẽ, liên tục kiểm tra Turn để xem khi nào nó trở thành 1. Việc kiểm tra liên tục một biến cho đến khi một số giá trị xuất hiện được gọi là chờ bận. Một khóa sử dụng chờ bận được gọi là khóa quay.

+Khi tiến trình 0 rời khỏi vùng quan trọng, Turn sẽ chuyển thành 1, để cho phép tiến trình 1 đi vào vùng quan trọng của nó

+Nếu tiến trình 0 cố gắng đi vào vùng quan trọng của nó một lần nữa, tiến trình đó sẽ bị chặn vì Turn không còn là 0 nữa.

Slide 10:

Tiếp theo là đề xuất phần mềm thứ 3 là Giải pháp của Peterson

Các tiến trình chia sẻ hai biến chung: turn và increase [2]

Nếu interesse[i] = TRUE có nghĩa là tiến trình Pi muốn vào vùng quan trọng. Khởi đầu, interesse[0]=interesse[1]=FALSE và giá trị của est được khởi động là 0 hay 1. Để có thể vào được vùng quan trọng, trước tiên tiến trình Pi đặt giá trị interesse[i]=TRUE ( xác định rằng tiến trình muốn vào vùng quan trọng), sau đó đặt turn=j (đề nghị thử tiến trình khác vào vùng quan trọng). Nếu tiến trình Pj không quan tâm đến việc vào vùng quan trọng (interesse[j]=FALSE), thì Pi có thể vào vùng quan trọng, nếu không, Pi phải chờ đến khi interesse[j]=FALSE. Khi tiến trình Pi rời khỏi vùng quan trọng, nó đặt lại giá trị cho interesse[i]= FALSE.

Slide 11:

Nhận xét về phương pháp của Peterson :

Thỏa mãn 3 điều kiện:

+ Loại trừ lẫn nhau

Pi có thể nhập vùng nhớ quan trọng khi interes[j] == F hoặc Turn==i

Nếu cả hai đều muốn quay lại, vì Turn chỉ có thể nhận giá trị 0 hoặc 1, vì vậy một quá trình nhập vùng nhớ quan trọng

+ Phát triển

Sử dụng 2 biến khác biệt interes[i] ==> không thể khóa đối lập

Chờ đợi có giới hạn: cả interes[i] và Turn thay đổi

+ Không mở rộng thành N quy trinh

Slice 12:

Bây giờ chúng ta sẽ đi tìm hiểu các đề xuất phần cứng, thứ nhất là Vô hiệu hóa các ngắt.

Có lẽ cách rõ ràng nhất để đạt được loại trừ lẫn nhau là cho phép một tiến trình vô hiệu hóa các ngắt trước khi nó đi vào vùng quan trọng và sau đó cho phép kích hoạt lại các ngắt sau khi nó rời khỏi vùng quan trọng. Bằng cách vô hiệu hóa ngắt, CPU sẽ không thể chuyển đổi các tiến trình. Điều này đảm bảo rằng tiến trình có thể sử dụng biến được chia sẻ mà không cần tiến trình khác truy cập vào nó.

Cách tiếp cận này nói chung là không hấp dẫn vì không khôn ngoan khi cung cấp cho các quy trình của người dùng quyền để tắt các ngắt. Giả sử rằng một trong số họ đã làm điều đó và không bao giờ bật chúng lên nữa? Đó có thể là sự kết thúc của hệ thống.Hơn nữa, nếu hệ thống là một bộ xử lý đa năng, có hai CPU trở lên, thì việc tắt ngắt chỉ ảnh hưởng đến CPU đã thực thi lệnh vô hiệu hóa. Những cái khác sẽ tiếp tục chạy và có thể truy cập bộ nhớ dùng chung.

Slide 13:

Giải pháp phần cứng tiếp theo là Chỉ thị TSL (Test-and-Set Lock)

CPU nguyên thủy hỗ trợ Kiểm tra và Đặt Khóa.

+ Trả về giá trị hiện tại của một biến, đặt biến thành giá trị true.

+ Việc kiểm tra, cập nhật nội dung một vùng nhớ trong một thao tác không thể phân chia

Để sử dụng lệnh TSL, chúng ta sẽ sử dụng một biến chia sẻ, lock, để điều phối quyền truy cập vào bộ nhớ được chia sẻ. Khi lock bằng 0, bất kỳ tiến trình nào cũng có thể đặt nó thành 1 bằng cách sử dụng lệnh TSL và sau đó đọc hoặc ghi bộ nhớ dùng chung. Khi nó được thực hiện, tiến trình đặt lock trở lại 0 bằng cách sử dụng một lệnh di chuyển thông thường.

Slide 14:

Nhận xét cho các giải pháp phần cứng trong Bận chờ

+Hỗ trợ cơ chế phần cứng cần thiết

Không dễ dàng với hệ thống nhiều CPU

+ Dễ dàng mở rộng đến N quy trình

+ Sử dụng CPU không hiệu quả

Liên tục kiểm tra tình trạng khi chờ vùng quan trọng

+ Quá tải

Khóa các tiến trình không đủ điều kiện để đi kèm trong vùng quan trọng, nhượng bộ CPU cho tiến trình khác

Sử dụng Trình lập lịch biểu

Chờ và xem...