1. Giải pháp nào có thể xảy ra tình huốn hai tiến trình trong đoạn gang cùng lúc

* Biến cờ hiệu
* Kiểm tra luân phiên
* **Peterson**
* Test&Set

1. Chọn phát biểu đúng về nhóm giải pháp sleep-wakeup

* Khi chưa đủ điều kiện vào đoạn gang, tiến trình chuyển sang terminated
* **Khi chưa đủ điều kiện vào đoạn gang, tiến trình chuyển sang trạng thái Waiting**
* **Tiến trình đang sleep cần một tiến trình khác đánh thức**
* Mọi phát biểu đều đúng

1. Đâu là các giải pháp busy-waiting

* **Biến cờ hiệu**
* Monitor
* **Peterson**
* **Kiểm tra luân phiên**

1. Điều kiện nào sau đây không cần thiết khi giải quyết bài toán đoạn găng

* Không có hai tiến trình đồng thời trong đoạn gang
* Không có tiến trình chờ vô hạn để vào đoạn gang
* **Phải giả thiết về tốc độ tiến trình và số lượng CPU**
* Tiến trình bên ngoài gang không được ngăn cản các tiến trình khác vào đoạn gang

1. Kỹ thuật nào được sử dụng để liên lạc trực tiếp giữa hai tiến trình

* Vùng nhớ chia sẻ
* **Đường ống**
* Trao đổi thông điệp
* Socket

1. Phương pháp nhanh nhất để chia sẻ dữ liệu giữa cac tiến trình

* Đường ống
* Truyền thông điệp
* **Vùng nhớ chia sẻ**
* Socket

1. Hai tiến trình chia sẻ một semaphore Dijkstra s, khởi gán e(s) là 0; hai tiến trình có cấu trúc như sau

P1:

While(TRUE){

Job1(); Up(s);

}

P2:

While(TRUE){

Down(s);

}

* **P2 chỉ thực hiện sau P1 thực hiện**
* Đoạn code trên giải quyết bài toán độc quyền truy xuất
* **Đoạn code trên giải quyết bài toán đồng bộ**
* Down(s) là thao tác Nguyên tử, Up(s) không là thao tác nguyên tử

1. Chọn phát biểu đúng về thao tác nguyên tử

* Không thể chia nhỏ thời gian sử dụng CPU khi thực hiện thao tác nguyên tử
* Tiến trình thứ hai có thể tranh CPU của tiến trình đang thực hiện thao tác nguyên tử
* Hai phát biểu đều đúng
* Hai phát biểu đều sai

1. Với giải pháp Test&Set, nếu theo tác Test&SetLock không là thao tác nguyên tử thì

* Không ảnh hưởng gì đến giải pháp
* Không cập nhật được biến lock
* Có thể tồn tại nhiều tiến trình đồng thời trong đoạn gang
* **Không thể cài đặt thuật toán**

1. Chọn phát biểu đúng về nhóm giải pháp busy-waiting

* **Các giải pháp busy-wating làm mất thời gian CPU vì luôn phải thực hiện kiểm tra khi chưa đủ điều kiện vào gang**
* Khi chưa đủ điều kiện vào găng, tiến trình luôn ở trạng thái running
* Khi chưa đủ điều kiện vào găng tiến trình ở ready hoặc running
* Khi chưa đủ điều kiện vào găng, tiến chuyển sang waiting

1. Trong giải pháp Test&Set, trong thời gian tiến trình P1 ở trong đoạn găng, có 10 tiến trình khác muốn vào găng và đã kiểm tra điều kiện vào găng. Biến lock được cập nhật bao nhiêu lần và giá trị cuối cùng là gì

* **0, true**
* 1, true
* 100, true
* 100, false

1. Chọn phát biểu không chính xác

* Các tiến trình có thể thông báo cho nhau về một sự kiện
* **Mỗi tiến trình xử lý tín hiệu theo cách riêng**
* Có thể đồng bộ các tiến trình trong Windows
* Các tiến trình có thể trao đổi dữ liệu thông qua vùng nhớ chia sẻ

1. Giải pháp Test&Set có giải quyết triệt để bài toán độc quyền truy xuất khi hệ thống sử dụng nhiều CPU

* **Không**
* Có

1. Chọn các phát biểu không chính xác về đoạn găng

* **Đoạn găng là đoạn mã lệnh dễ xảy ra tranh chấp tài nguyên khi tiến trình sử dụng nó**
* **Đoạn găng là đoạn dữ liệu tranh chấp của chương trình**
* Đoạn găng là tập các file độc quyền truy xuất
* **Cả 3 phát biểu trên đều đúng**

1. Chọn phát biểu đúng về giải pháp Semaphore của Dijkstra

* Khi giải quyết bài toán độc quyền truy xuất e(s) cần khởi tạo = 1
* **Khi giải quyết bài toán đồng bộ e(s) cần khởi tạo = 1**
* Khi giải quyết bài toán độc quyền truy xuất cần khởi tạo = 0
* Khi giải quyết bài toán đọc quyền try xuất e(s) không cần khởi tạo

1. Tiến trình chỉ có trạng thái running khi ở trong đoạn găng

* **Đúng**
* Sai

1. Test-and-Setlock(Boolean target)

{ Boolean temp = target;

Target = TRUE;//thiết lập giá trị mới = true để khóa

Return temp;//lấy giá trị cũ để kiểm tra

Cài đặt tiến trình:

While(TRUE){

While(Test-and-Setlock(lock)){//wait}

Critical-section();lock = False;

Lock=False(không khóa) Noncritical-section();

}

* Chọn phát biểu đúng
* **Test-and-Setlock(boolean target) là thao tác nguyên tử**
* Đoạn mã giải quyết bài toán độc quyền truy xuất
* Vẫn có tình huống hai tiến trình có mặt đồng thời trong đoạn găng nếu sử dụng đoạn mã trên

1. Với giải pháp Semaphone Dijkstra, tất cả các tiến trình được cài đặt như sau:

While(TRUE){

Down(s)

Cirtical-section();

Up(s)

Noncritical-section();

}

Và e(s) khởi tạo = 1.

Sử dụng để làm gì?

* Giải quyết vấn đề độc quyền truy xuất
* Đảm bảo không có nhiều tiến trình có mặt đồng thời trong đoạn găng
* **Để đồng bộ giữa các tiến trình**
* Để liên lạc tiến trình

1. Chọn giải pháp phần cứng để bảo vệ đoạn găng

* Biến cờ hiệu
* Cấm ngắt
* Kiểm tra luân phiên
* Peterson
* **Test&Set**

1. Chọn phát biểu sai về busy-waiting

* Tất cả các giải pháp trên đều phải thực hiện một vòng lặp để kiểm tra xem có được phép vào đoạn găng
* Nếu tiến trình chưa cho phép, tiến trình phải chờ tiếp tục trong vòng lặp kiểm tra này
* Vòng lặp kiểm tra điều kiện vào găng nói chung: while(!điều kiện vào găng){}
* **Cả 3 phát biểu trên đều sai**
* Chỉ có hai phát biểu đúng