**SEMAPHORE**

1. **Giới thiệu**

Mối tương quan về tốc độ thực hiện của các tiểu trình trong hệ thống là không thể biết trước. Vì điều này phụ thuộc vào yếu tố động như tần suất xảy ra ngắt của từng tiểu trình, thời gian tiểu trinh được cấp phát bộ nhớ xử lý, … Có thể nói rằng các tiểu trình hoạt động không đồng bộ với nhau. Nhưng có những tình huống các tiểu trình cần hợp tác với nhau trong việc hoàn thành tác vụ, khi đó cần phải đồng bộ hóa hoạt động của các tiểu trình. Ví dụ một tiểu trình chỉ có thể xử lý nếu một tiểu trình khác đã kết thúc một công việc nào đó.

1. **Bài toán đồng bộ hóa**
2. Tranh đoạt điều khiển (race condition)

Giả sử có hai tiểu trình P1 và P2 cùng thực hiện công việc rút tiền, cùng chia sẻ vùng nhớ dùng chung lưu trữ biến *taikhoan* (số tiền hiện có trong tài khoản). Mỗi tiểu trình sẽ thực hiện rút một số tiền *tienrut* từ tài khoản:

if (taikhoan – tienrut > 0)

taikhoan = taikhoan – tienrut;

else

printf (“Khong the rut tien!”);

Giả sử trong tài khoản hiện còn 900, P1 muốn rút 600 và P2 muốn rút 400. Nếu xảy ra tình huống như sau:

* Sau khi đã kiểm tra điều kiện (taikhoan – tienrut > 0) và nhận kết quả là 300, P1 hết thời gian xử lý mà hệ thống cho phép, hệ điều hành cấp phát CPU cho P2.
* P2 kiểm tra cùng điều kiện trên và nhận được kết quả là 500 (do P1 vẫn chưa rút tiền) và rút 400. Giá trị *taikhoan* được cập nhật là 500.
* Khi P2 được tái kích hoạt và tiếp tục xử lý thì nó sẽ bỏ qua việc kiểm tra điều kiện *(taikhoan – tienrut > 0)*, vì đã kiểm tra trong lượt xử lý trước. P2 tiếp tục rút với số tiền là 600. Lúc này *taikhoan* sẽ được cập nhật lại là -100 🡪 Tình huống lỗi xảy ra.

Các tình huống lỗi tương tự như trên có thể xảy ra nếu có hai hay nhiều tiểu trình cùng đọc và ghi trên một vùng nhớ dùng chung, và kết quả cuối cùng phụ thuộc vào sự điều phối của hệ thống. Trường hợp này được gọi là các tình huống tranh đoạt điều khiển (race condition).

1. Miền găng (critical section)

Để ngăn chặn các tình huống lỗi có thể phát sinh khi các tiểu trình truy xuất đồng thời một tài nguyên không thể chia sẻ, cần phải đặt một sự truy xuất độc quyền cho tài nguyên đó: khi một tiểu trình đang sử dụng tài nguyên, thì những tiểu trình khác không được truy xuất đến tài nguyên đó.

Đoạn chương trình trong đó có khả năng xảy ra các mâu thuẫn truy xuất trên tài nguyên dùng chung được gọi là miền gang (critical section). Trở lại ví dụ trên, mỗi đoạn mã sau đây của mỗi tiểu trình tạo thành một miền găng:

if (taikhoan – tienrut > 0)

taikhoan = taikhoan – tienrut;

Có thể giải quyết vấn đề mâu thuẫn truy xuất nếu có thể đảm bảo tại một thời điểm chỉ có duy nhất một tiểu trình được xử lý lệnh trong miền găng.

Một phương pháp giải quyết tốt bài toán miền gang cần thỏa mãn 4 điều kiện sau:

* Không có hai tiểu trình cùng ở trong miền găng cùng lúc.
* Không có giả thiết nào đặt ra cho sư liên hệ về tốc độ của các tiểu trình, cũng như về số lượng bộ xử lý trong hệ thống.
* Một tiểu trình tạm dừng bên ngoài miền găng không được ngăn cãn các tiểu trình khác vào miền găng.
* Không có tiểu trình nào phải chờ vô hạn để được vào miền găng.

1. **Biên dịch**

gcc –o example example.c –lpthread