**MULTI THREAD TRONG LINUX**

1. **Process**

**Process là gì?**

Trong UNIX chuẩn thì định nghĩa là: một không gian địa chỉ với một hoặc nhiêu các thread hoạt động trong không gian địa chỉ đó.

**Process in memory**

**Từng process sẽ được phân bổ cho một số duy nhất gọi là process identifier or PID. Nó thường là một số nguyên dương 2-32.768. Khi một process được bắt đầu, số tiếp theo chưa sử dụng sẽ được chọn. Số 1 đặc biệt sẽ tiến trình quản lý các tiến trình khác.**

**Các cách khởi tạo process**

Có thể khởi tạo bằng thư viện system . Trong ví dụ của sách thì hàm system

Ta cũng có thể dụng thư viện <sys/types.h>. Để khởi tạo một process theo cú pháp: pid\_t name. name=name\_process();

**So sánh Process và thread**

Thread nằm trong các process. Nó sử dụng chung không gian bộ nhớ của process. Nó sẽ được thực hiện thông qua một lịch trình < thread scheduler>.

1. **Thread là gì?**

Trong mỗi process lại chứa nhiều các thread, các thread này sử dụng chung vùng nhớ trong toàn bộ phạm vi của process. Ta có thể coi một thread là một process con nhưng chạy trên cùng một không gian bộ nhớ.

1. **Tạo ra các thread**

**Ta có thể sử dụng thư viện pthread.h.**

# **4. Terminate thread**

**Đồng bộ truy cập data giữa các thread trong một process**

**Trong các phần trước chúng ta đã đề cập thời các thread thực hiện cùng nhau. Nhưng phương th đảo giữa chúng là một phương (clumsy and very inefficient) vụng về và không hiệu quả. Có hai phương thức cơ bản thường dùng là semaphore và mutexes.**

## **4.1. Đồng bộ sử dụng semaphore**

**Semaphore là một kiểu biến đặc biết có thể được tăng hoặc giảm. Nó đảm bảo rằng tại một thời điểm thì chỉ có một thread try cập vào được critical section< read file, write file..>.**

**Trong C, ta có thể sử dụng thư viện semaphore.h.**

## **4.2. Đồng bộ sử dụng mutex**

**Mutex có thể hiểu đơn giản là loại trừ lẫn nhau. Loại trừ lẫn nhau có thể hiểu là chỉ một luồng có thể truy cập vào tài nguyên nào đó. Đối tượng mutex cho phép nhiều luồng chương trình sử dụng cùng một tài nguyên nhưng mỗi lần một lường không đồng thời sử dụng.**

## **4.3. So sánh Mutex và Semaphore**

**+Semaphore được hiểu là cơ chế báo hiệu, còn mutex là cơ chế khóa.**

**+Semaphore là một số nguyên còn mutex là một đối tượng.**

**Kết luận: Semaphore là một lựa chọn tốt hơn trong trường hợp có nhiều phiên bản tài nguyên có sắn. Trong trường hợp mutex tài nguyên chia sẻ duy nhất là một lựa chọn tốt hơn.**

1. **Thay đổi các giá trị của một thread**

1. **Điều khiển thread từ một thread khác trong cùng một process**

***Đôi khi ta cũng muốn một tuyến khác diều khiển một tuyến khác trong process. Trong tài liệu này tôi chỉ đề cập tới một số cách như là khỏi tạo một tuyến bằng một tuyến khác, chấm dứt một tuyến khác. Ta có thể sử dụng hàm pthread\_cancl() để gửi tín hiệu đến tuyến cần hủy.***