Câu 1:

**Fork:**

* Tạo process mới, là con của process hiện tại đang gọi
* Cả 2 process đều thực hiện lệnh tiếp theo sau lệnh fork
* 2 process dùng riêng các tài nguyên của máy tính như không gian địa chỉ, code,…

**Thread**:

* Tạo thread mới giống với process đang gọi
* Thread trong cùng 1 process dùng chung tài nguyên bộ nhớ
* Thread mới dùng riêng địa chỉ stack, ID, thanh ghi

Fork và thread đều clone task, fork ít chia sẻ tài nguyên giữa process mới và process được clone hơn pthread\_create

Nên dùng fork khi thực hiện task phức tạp hoặc muốn xử lý các task tách biệt với nhau, dùng pthread\_create khi muốn xử lý subtask đơn giản.

Câu 2:

**Định nghĩa của Semaphore**

Semaphore là một công cụ đồng bộ hóa quá 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 quá trình đang sửa đổi giá trị của semaphore, không có quá 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 quá 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 quá 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 quá 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 quá 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, quá 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, quá 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.