1. ***So sánh chi tiết lại điểm mạnh và điểm yếu giữa  fork và pthread, nêu lên các trường hợp ứng dụng nên sử dụng fork và pthread\_create?***

**So sánh:**

* Fork():
  + Fork tạo ra một process con (child process) là copy (nhân bản) của chính process tạo ra nó (parent process)
  + Cả 2 process thực hiện lệnh tiếp theo sau lệnh gọi fork()
  + Bản sao và gốc giống hệt về không gian địa chỉ, code và stack
* Pthread\_create():
  + Tạo một luồng (thread) mới trong process.
  + Các luồng trong 1 process có thể giao tiếp bằng bộ nhớ dùng chung
  + Luồng mới sẽ nhận được id và stack riêng

**Các trường hợp sử dụng fork và pthread\_create:**

* Fork():

Trong các máy chủ web server(ví dụ apache) sử dụng fork để tạo ra các tiến trinh độc lập. Mỗi process xử lý các yêu cầu trong không gian địa chỉ riêng. Nếu một bộ nhớ bị chết/ không hoạt động thì không ảnh hưởng tới process còn lại.

* Pthread\_create():
* Vì sử dụng chia sẻ dữ liệu và thời gian thực hiện tốt hơn nên phù hợp cho các ứng dụng real time. Ví dụ như đọc báo – hỗ trợ comment hiển thị trực tuyến.

1. ***Trình bày chi tiết Semaphore, và Mutex***

* Semaphore là một cơ chế **báo hiệu** khi hoạt động Wait () và signal () được thực hiện trên biến semaphore cho biết liệu một quá trình có thu được tài nguyên hay giải phóng tài nguyên hay không. Mặt khác, mutex là một cơ chế **khóa**, vì để có được một tài nguyên, một quá trình cần phải khóa đối tượng mutex và trong khi phát hành một quy trình tài nguyên phải mở khóa đối tượng mutex.
* Semaphore thường là một biến **số nguyên** trong khi đó, mutex là một **đối tượng** .
* Semaphore cho phép nhiều luồng chương trình truy cập vào **tài nguyên hữu hạn** . Mặt khác, Mutex cho phép nhiều luồng chương trình truy cập vào một **tài nguyên được chia sẻ** nhưng mỗi lần một **tài nguyên** .
* Giá trị biến semaphore có thể được sửa đổi bởi **bất kỳ** quá trình nào có được hoặc giải phóng tài nguyên bằng cách thực hiện thao tác Wait () và signal (). Mặt khác, khóa thu được trên đối tượng mutex chỉ có thể được giải phóng bằng quá trình đã thu được khóa trên đối tượng mutex.
* Semaphore có hai loại **đếm semaphore và semaphore nhị phân** khá giống với mutex.
* Giá trị biến semaphore được sửa đổi bằng thao tác **Wait ()** và **signal ()** ngoài khởi tạo. Tuy nhiên, đối tượng tắt tiếng bị khóa hoặc mở khóa bằng quá trình lấy hoặc giải phóng tài nguyên.
* Nếu tất cả các tài nguyên được thu thập bởi quy trình và không có tài nguyên nào là miễn phí thì quy trình mong muốn có được tài nguyên thực hiện thao tác Wait () trên biến semaphore và tự **chặn** cho đến khi số lượng semaphore lớn hơn 0. Nhưng nếu một đối tượng mutex đã bị khóa sau đó quá trình mong muốn có được tài nguyên **chờ đợi** và được hệ thống **xếp hàng** cho đến khi tài nguyên được giải phóng và đối tượng mutex được mở khóa.