1. What are the possible state transitions of a process?

Tiến trình có thể ở bất kỳ một trong ba trạng thái sau đây:

+ Running ( Đang sử dụng CPU tại thời điểm đó và đang chạy).

+ Ready ( Tạm thời dừng lại lại để cho phép một tiến trình khác chạy).

+ Blocked ( Không chạy cho đến khi có một sự kiện nào đó xảy ra ).

1. What are the differences between a thread and a process?

Định nghĩa:

+ Tiến trình: có nghĩa là một chương trình đang được thựchiện.

+ Luồng: có nghĩa là một phân đoạn của một tiến trình.

- bộ nhớ :

+ Tiến trình: Nặng

+ Luồng: nhẹ hơn tiến trình nhiều

- Thời gian dừng:

+ Tiến trình: mất nhiều thời gian hơn để chấm dứt.

+ Luồng: mất ít thời gian hơn để chấm dứt.

- Thời gian tạo:

+ Tiến trình: Cần nhiều thời gian hơn

+ Luồng: mất ít thời gian hơn- Tài nguyên:

+ Tiến trình: tiêu thụ nhiều tài nguyên hơn.

+ Luồng: tiêu thụ ít tài nguyên hơn.

- Chia sẻ:

+ Tiến trình: Nó không chia sẻ dữ liệu

+ Luồng: chia sẻ dữ liệu với nhau.

- Memory:

+ Tiến trình: chủ yếu bị cô lập

+ Luồng: chia sẻ bộ nhớ

1. User-level threads vs. Kernel-level threads? Know the pros and cons of each type.

Không có đặc quyền kernel mode cần thiết để chuyển đổi luồng trong các User-level threads.

+ Nhược điểm của User-Level Threads: Các ứng dụng đa luồng trong user-level threads không thểsử dụng đa xử lý để làm lợi thế của chúng. Toàn bộ quá trình bị chặn nếu một user-level threads thực hiện thao tác chặn.

+ Ưu điểm của Kernel-Level Threads: Nhiều luồng của cùng một quá trình có thể được lập lịch trên các bộ xử lý khác nhau trong Kernel-Level Threads. Các Kernel-Level Threads cũng có thể được đa luồng. Nếu Kernel-Level Threads bị chặn, nhân có thể lên lịch một luồng khác của cùng một tiến trình.

+ Nhược điểm của Kernel-Level Threads: Cần chuyển chế độ sang kernel mode để chuyển quyền điều khiển từ luồng này sang luồng khác trong một quy trình. Kernel-level threads chậm hơn để tạo cũng như quản lý sovới user-level threads

4) Understand the concepts of race condition and mutual exclusion

+ Locked variables, strict alternation, Peterson's algorithm, disabling interrupts, semaphore, monitor. Know which one is software or hardware solution; which one relies on busy waiting.

+ Semaphores. Understand what a semaphore is; its up and down operations; its principles to allow mutual exclusion among several processes or threads.

+ What is the preemption and non-preemption in the context of process scheduling?

+ How the FIFO, SJF (SJRF) and Round-Robin scheduling algorithms work?