CHƯƠNG 3: PROCESSES

Bài 1: Phân Tán Tính Toán bằng Process Pool

- Sử dụng Process Pool để thực hiện tính toán song song trong hệ thống phân tán.
- Chia nhỏ một tác vụ lớn thành nhiều phần và phân phối chúng cho nhiều tiến trình xử lý đồng thời.
- So sánh hiệu suất giữa việc chạy trên một tiến trình đơn và nhiều tiến trình song song.

Áp dụng

a) Tính tổng của 1 triệu số bằng cách chia nhỏ danh sách và chạy trên nhiều tiến trình

b) Cho phép người dùng chọn số tiến trình chạy song song.

c) Áp dụng tính tổng các số nguyên tố trong khoảng lớn.

d) Thử nghiệm với bộ dữ liệu lớn và đo thời gian thực thi.

e) So sánh giữa Process Pool và Thread Pool để thấy sự khác biệt.

f) Chạy trên hệ thống phân tán thật sự: Chia nhỏ dữ liệu và gửi cho nhiều máy khác nhau xử lý.

Bài 2: Giải quyết Deadlock bằng Timeout

- Mô phỏng tình huống deadlock giữa các tiến trình khi truy cập tài nguyên.
- Sử dụng timeout để phát hiện và xử lý deadlock.
- Tránh deadlock bằng cách giải phóng tài nguyên sau một thời gian nhất định nếu không lấy được khóa (lock).

a) Tạo 2 tiến trình (Process A và Process B). Mỗi tiến trình giữ một tài nguyên (Resource 1, Resource 2) và chờ tài nguyên còn lại. Nếu cả hai tiến trình đều giữ tài nguyên của mình và chờ vô hạn tài nguyên của tiến trình kia → xảy ra Deadlock. Dùng timeout để thoát khỏi deadlock nếu chờ quá lâu.

b) Kiểm tra nếu có deadlock thì hủy bỏ tiến trình hoặc giải phóng tài nguyên.

c) Thêm nhiều tiến trình và nhiều tài nguyên hơn để kiểm tra tình trạng deadlock phức tạp hơn.

d) Sử dụng Banker’s Algorithm để phát hiện trước deadlock và ngăn chặn.

e) Dùng mutex hoặc semaphore để quản lý tài nguyên tốt hơn.

Bài 3: Đồng bộ hóa nhiều luồng với Semaphore

- Hiểu và sử dụng Semaphore để quản lý truy cập tài nguyên dùng chung.
- Hạn chế số lượng luồng truy cập đồng thời vào một tài nguyên nhất định.
- Áp dụng Semaphore vào bài toán mô phỏng nhiều luồng truy cập vào một hệ thống giới hạn (ví dụ: giới hạn số người vào phòng, số kết nối vào server, v.v.).

a) Viết chương trình có n luồng (n=10) đại diện cho n người dùng cố gắng truy cập vào một tài nguyên chung (ví dụ: vào phòng học, vào máy in, truy cập server). Chỉ cho phép tối đa k luồng (k=3) chạy đồng thời (giống như chỉ có k chỗ trống). Nếu số luồng đang chạy đạt đến giới hạn k, các luồng khác phải chờ đến khi có chỗ trống. Khi một luồng kết thúc công việc, nó sẽ trả lại chỗ trống, cho phép luồng khác tiếp tục. Chương trình in ra log để hiển thị thứ tự truy cập của các luồng.

b) Viết chương trình trong đó nhiều luồng ghi vào một file, nhưng chỉ một luồng có thể ghi vào cùng lúc để tránh xung đột.

c) Mô phỏng một hệ thống quản lý truy cập phòng họp, chỉ cho phép tối đa N người vào cùng lúc. Khi phòng họp đã đầy, các nhân viên khác phải chờ đến lượt.

Bài 4: Xử lý Tranh Chấp Dữ Liệu Giữa Các Luồng (Race Condition)

- Hiểu và mô phỏng tranh chấp dữ liệu (race condition) khi nhiều luồng cùng truy cập và cập nhật một tài nguyên dùng chung.
- Sử dụng Lock/Mutex để giải quyết vấn đề truy cập đồng thời.
- So sánh chạy không đồng bộ (có lỗi race condition) và chạy đồng bộ (không có lỗi nhờ Lock).

a) Mô phỏng race condition: Tạo nhiều luồng cùng cập nhật một biến đếm chung counter (Nếu không dùng Lock, giá trị counter sẽ không chính xác vì nhiều luồng cùng ghi vào biến này cùng lúc).

b) Giải quyết race condition bằng Lock: Dùng Lock (mutex) để đảm bảo chỉ một luồng có thể cập nhật counter tại một thời điểm. So sánh kết quả giữa chạy không đồng bộ (không có Lock) và chạy đồng bộ (có Lock).

c) Dùng Rlock (Reentrant Lock) để tránh Deadlock

Bài 5: Giao Tiếp Giữa Các Tiến Trình Bằng Pipe

- Hiểu cách giao tiếp giữa các tiến trình (IPC - Inter-Process Communication) bằng pipe.
- Biết cách truyền dữ liệu từ tiến trình cha sang tiến trình con hoặc ngược lại.
- Ứng dụng pipe để chia sẻ thông tin giữa các tiến trình mà không cần file trung gian.

a) Từ tiến trình cha tạo một tiến trình con. Sử dụng pipe để gửi dữ liệu từ tiến trình cha sang tiến trình con. Tiến trình con đọc dữ liệu từ pipe và hiển thị kết quả ra màn hình. Đảm bảo tiến trình con chỉ đọc sau khi cha đã ghi xong để tránh lỗi đồng bộ.

b) Gửi dữ liệu từ tiến trình con về tiến trình cha (tạo thêm pipe thứ hai).

c) Giao tiếp hai chiều giữa tiến trình cha và con.

d) Nhiều tiến trình con gửi dữ liệu đến cha. Tiến trình cha đọc và xử lý từng dữ liệu nhận được.