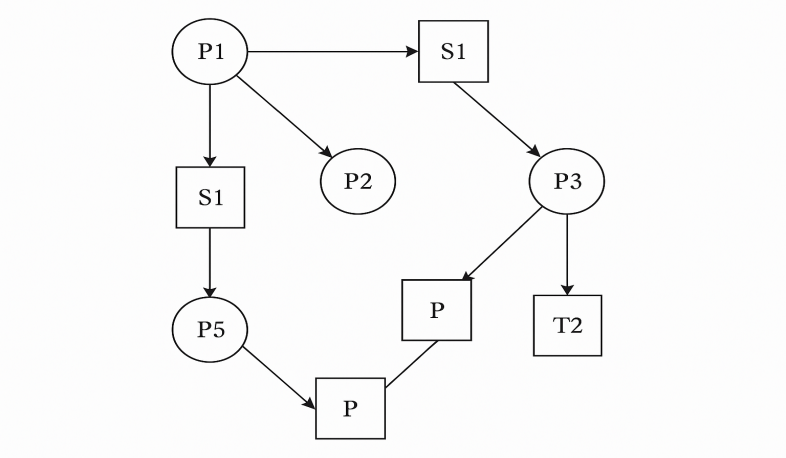
1. Một hệ thống có 2 máy quét hình, 1 máy in và 2 ổ băng từ. Cho 5 tiến trình P1, P2, P3, P4 và P5 đang vận hành với trạng thái cấp phát tài nguyên như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Tiến trình | Máy quét hình | Ổ băng từ | Máy in |
| P1 | Yêu cầu | Được cấp 1 phiên bản |  |
| P2 | Được cấp 1 phiên bản |  | Được cấp 1 phiên bản |
| P3 | Được cấp 1 phiên bản | Yêu cầu |  |
| P4 |  | Được cấp 1 phiên bản | Yêu cầu |
| P5 | Yêu cầu |  | Yêu cầu |

Hãy:

1. Hãy vẽ biểu đồ RAG



1. Xác định và giải thích trạng thái cấp tài nguyên trên có Deadlock?

P1 giữ T1, đòi máy quét hình.

P3 giữ S2, đòi ổ băng từ.

P4 giữ T2, đòi máy in.

P2 giữ S1 và máy in (P).

P5 yêu cầu máy quét hình và máy in.

P1 giữ T1, đợi máy quét hình (bị P2/P3 giữ).

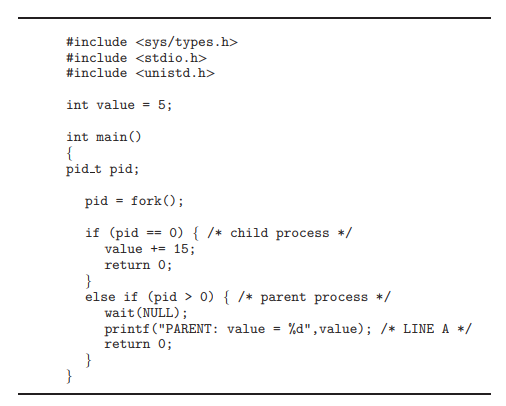
P3 giữ S2, đợi ổ băng từ (bị P1 và P4 giữ).

P4 giữ T2, đợi máy in (bị P2 giữ).

P2 đang giữ S1 và máy in.

P5 đang đợi máy quét hình và máy in (cũng liên quan tài nguyên đang bị chiếm).

=> Có vòng lặp => Deadlock tồn tại

1. Sử dụng chương trình trong Hình 1, giải thích kết quả đầu ra sẽ như thế nào? ở LINE A

Hình 1

fork() tạo ra một tiến trình con.

Tiến trình con sẽ thực hiện value += 15; (tăng value thêm 15) rồi kết thúc.

Tiến trình cha sẽ wait(NULL); (đợi tiến trình con kết thúc), sau đó in ra giá trị value.

Và Biến value được lưu trong bộ nhớ riêng của từng tiến trình sau fork().

Vì thế, tiến trình con thay đổi value sẽ không ảnh hưởng đến tiến trình cha.

- Giá trị value được in ra sẽ là 5

1. Giải thích sự khác biệt giữa việc điều phối thời gian tiếm quyền (preemptive) và không tiếm quyền (nonpreemptive).

- Preemptive (tiếm quyền): Hệ thống có thể tạm dừng tiến trình đang chạy và chuyển quyền điều khiển cho tiến trình khác bất cứ lúc nào, ví dụ như khi hết time quantum. Điều này giúp hệ thống phản hồi nhanh hơn và tăng tính công bằng giữa các tiến trình, nhưng có thể gây overhead cao hơn do việc chuyển đổi giữa các tiến trình.

- Nonpreemptive (không tiếm quyền): Khi một tiến trình đang chạy, hệ thống không thể tạm dừng nó trừ khi tiến trình đó tự ngừng (ví dụ, kết thúc hoặc đợi tài nguyên). Điều này giúp giảm overhead và cải thiện hiệu suất cho các tiến trình dài, nhưng có thể dẫn đến sự thiếu công bằng khi một tiến trình có thể chiếm hết tài nguyên.

1. Lợi ích gì khi có nhiều time-quantum khác nhau ở các mức độ khác nhau của hệ thống hàng chờ phân cấp?

Khi có nhiều mức time quantum khác nhau trong hệ thống hàng chờ phân cấp, mỗi mức sẽ phục vụ các tiến trình với độ ưu tiên và yêu cầu khác nhau. Điều này mang lại một số lợi ích:

* **Cải thiện hiệu quả**: Các tiến trình ngắn và có độ ưu tiên cao sẽ không bị chậm lại bởi các tiến trình dài, trong khi các tiến trình dài sẽ có đủ thời gian để thực hiện.
* **Tăng tính công bằng**: Các tiến trình với độ ưu tiên thấp không bị trì hoãn quá lâu vì chúng được phân bổ thời gian CPU ngắn hơn.
* **Tối ưu hóa tài nguyên**: Tùy thuộc vào mức độ quan trọng của tiến trình, hệ thống có thể phân bổ thời gian CPU hợp lý cho các tiến trình khác nhau.

1. Thuật toán Banker trong hệ điều hành là gì? Tại sao thuật toán của Banker được đặt tên như vậy?

**Thuật toán Banker** là một thuật toán được sử dụng để tránh deadlock trong hệ điều hành. Nó kiểm tra xem liệu các yêu cầu tài nguyên của các tiến trình có thể được đáp ứng mà không gây ra deadlock hay không. Thuật toán này hoạt động như sau:

* **Kiểm tra tài nguyên hiện có và yêu cầu**: Trước khi cấp tài nguyên cho tiến trình, thuật toán kiểm tra liệu có thể cung cấp tài nguyên mà không làm hệ thống rơi vào tình trạng deadlock hay không. Nếu có thể, tài nguyên được cấp; nếu không, tiến trình phải đợi.

Thuật toán này được đặt tên theo khái niệm của một **ngân hàng**, nơi mỗi người chỉ được cấp tiền nếu ngân hàng có đủ tiền để đảm bảo rằng sau khi giao dịch, mọi người đều có thể nhận lại tiền của mình mà không làm cho hệ thống bị phá sản. Tương tự, trong hệ thống máy tính, tài nguyên được cấp phát chỉ khi có thể đảm bảo rằng không có tình trạng deadlock xảy ra trong tương lai.