**MÔN HỌC: HỆ ĐIỀU HÀNH**

**CÂU HỎI VÀ BÀI TẬP CHƯƠNG 6**

1. Deadlock là gì?

* Trả lời:
* Một tiến trình gọi là deadlock nếu nó đang đợi một sự kiện mà sẽ không bao giờ xảy ra.
* Một tiến trình gọi là trì hoãn vô hạn định nếu nó bị trì hoãn một khoảng thời gian dài lặp đi lặp lại trong khi hệ thống đáp ứng cho những tiến trình khác

1. Các điều kiện cần để xảy ra deadlock?

* Trả lời:
* Loại trừ tương hỗ: ít nhất một tài nguyên được giữ theo nonsharable mode.
* Giữ và chờ cấp thêm tài nguyên: Một tiến trình đang giữ ít nhất một tài nguyên và đợi thêm tài nguyên do tiến trình khác giữ.
* Không trưng dụng: tài nguyên không thể bị lấy lại mà chỉ có thể được trả lại từ tiến trình đang giữ tài nguyên đó khi nó muốn
* Chu trình đợi: tồn tại một tập (P0,…,Pn} các tiến trình đang đợi sao cho:
* P0 đợi một tài nguyên mà P1 giữ
* • P1 đợi một tài nguyên mà P2 giữ
* • …
* • Pn đợi một tài nguyên mà P0 giữ

1. Đồ thị cấp phát tài nguyên là gì? Mối liên hệ giữa đồ thị cấp phát tài nguyên và deadlock?

* Đồ thị cấp phát tài nguyên:
  1. Là đồ thị có hướng, với tập đỉnh V và tập cạnh E.
  2. Tập đỉnh V gồm 2 loại:
     1. P = {P1, P2,…,Pn} (All process)
     2. R = {R1, R2,…,Rn} (All resource)
  3. Tập cạnh E gồm 2 loại:
     1. Cạnh yêu cầu: Pi -> Rj
     2. Cạnh cấp phát: Rj-> Pi
* Liên hệ với Deadlock:
  1. RAG không chứa chu trình -> không có deadlock.
  2. RAG chứa một (hay nhiều) chu trình
     1. Nếu mỗi loại tài nguyên chỉ có một thực thể -> Deadlock.
     2. Nếu mỗi loại tài nguyên có nhiều thực thể -> có thể xảy ra Deadlock.

1. Có mấy phương pháp để giải quyết deadlock? Phân tích và đánh giá ưu, nhược điểm của từng phương pháp?

* Trả lời:
* Có 4 phương pháp giải quyết Deadlock:
  1. Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock và phục hồi hệ thống.
  2. Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống:
     1. Deadlock không được phát hiện, dẫn đến việc giảm hiệu suất của hệ thống. Cuối cùng, hệ thống có thể ngưng hoạt động và phải khởi động lại.
     2. Tránh deadlock: các tiến trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp.
     3. Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock và phục hồi hệ thống.
     4. Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống:
        + Deadlock không được phát hiện, dẫn đến việc giảm hiệu suất của hệ thống. Cuối cùng, hệ thống có thể ngưng hoạt động và phải khởi động lại.

1. Phân tích và đánh giá ưu, nhược điểm của các giải pháp đồng bộ busy waiting (cả phần cứng và phần mềm)?

* Trả lời:

1. Trạng thái an toàn là gì? Mối liên hệ giữa trạng thái an toàn và deadlock?

* Trả lời:
* Định nghĩa trạng thái an toàn:
  1. Một trạng thái của hệ thống được gọi là an toàn (safe) nếu tồn tại một chuỗi thứ tự an toàn.
  2. Một chuỗi tiến trình <P1, P2,…,Pn> là một chuỗi an toàn nếu
     1. Với mọi i = 1, …, n yêu cầu tối đa về tài nguyên của Pi có thể được thỏa bởi
        + Tài nguyên mà hệ thống đang có sẵn sàng.
        + Cùng với tài nguyên mà tất cả các Pj (j < i) đang giữ.
* Liên hệ với deadlock:
  1. Nếu hệ thống đang ở trạng thái safe -> không deadlock.
  2. Nếu hệ thống đang ở trạng thái unsafe -> có thể dẫn đến deadlock.
  3. Tránh deadlock bằng cách bảo đảm hệ thống không đi đến trạng thái unsafe.

1. Mô tả cách thực hiện các giải thuật Banker: giải thuật an toàn, giải thuật yêu cầu tài nguyên và giải thuật phát hiện deadlock?

* Trả lời:
* Giải thuật an toàn:
  1. Gọi Work và Finish là hai vector độ dài là m và n. Khởi tạo
     1. Work = Available
     2. Finish[i] = false, i = 0, 1, ..., n – 1
  2. Tìm i thỏa
     1. Finish[i] = false
     2. Needi <= Work (hàng thứ i của Need)
     3. Nếu không tồn tại i như vậy, đến bước 4.
  3. Work = Work + Allocationi
     1. Finish[i] = true
     2. Quay về bước 2
  4. Nếu Finish[i] = true, i = 1,...,n thì hệ thống đang ở trạng thái safe.
* Giải thuật yêu cầu tài nguyên:
  1. Requesti[j] = k ⬄ Pi cần k instance của tài nguyên Rj
  2. Nếu Requesti ≤ Needi thì đến bước 2. Nếu không, báo lỗi vì tiến trình đã vượt yêu cầu tối đa.
  3. Nếu Requesti ≤ Available thì qua bước 3. Nếu không, Pi phải chờ vì tài nguyên không còn đủ để cấp phát.
  4. Giả định cấp phát tài nguyên đáp ứng yêu cầu của Pi bằng cách cập nhật trạngthái hệ thống như sau:
     1. Available = Available – Requesti
     2. Allocationi = Allocationi + Requesti
     3. Needi = Needi – Requesti
* Giải thuật phát hiện deadlock
  1. Gọi Work và Finish là vector kích thước m và n. Khởi tạo:
     1. Work = Available
     2. For i = 1, 2,…, n, nếu Allocationi ≠ 0 thì Finish[ i ] := false

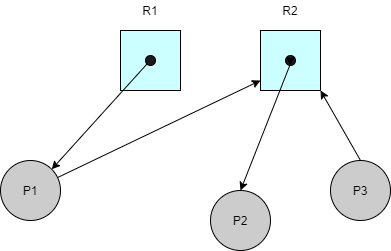
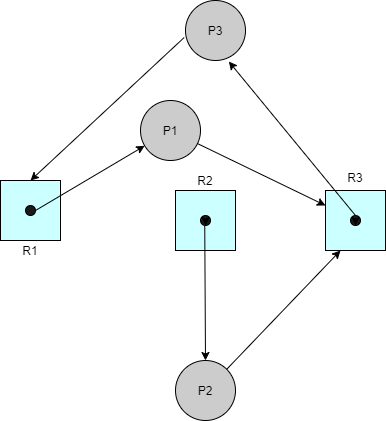
còn không thì Finish[ i ] := true

* 1. Tìm i thỏa mãn:
* Finish[ i ] = false
* Requesti ≤ Work
* Nếu không tồn tại i như vậy, đến bước 4.
  1. Work = Work + Allocation
     1. Finish[ i ] = true
     2. Quay về bước 2.
  2. Nếu Finish[ i ] = false, với một số i = 1,…, n, thì hệ thống đang ở trạng thái deadlock. Hơn thế nữa, Finish[ i ] = false thì Pi bị deadlocked.

1. Nêu các giải pháp để phục hồi hệ thống sau khi phát hiện có deadlock?

* Trả lời:
* Giải pháp:
  1. Báo người vận hành
  2. Hệ thống tự động phục hồi bằng cách bẻ gãy chu trình deadlock:
     1. Chấm dứt một hay nhiều tiến trình
     2. Lấy lại tài nguyên từ một hay nhiều tiến trình

1. (Bài tập mẫu) Cho các đồ thị cấp phát tài nguyên sau. Hỏi đồ thị nào có deadlock xảy ra?

(a) (b)

Trả lời:  
- Đồ thị (a) không có deadlock, do đồ thị này có chuỗi an toàn là: <P2, P1, P3> hoặc <P2, P3, P1>.

- Đồ thị (b) có deadlock.

1. (Bài tập mẫu) Cho 1 hệ thống có 4 tiến trình P1, P2, P3, P4 và 3 loại tài nguyên R1 (3), R2 (2) R3 (2). P1 giữ 1 R1 và yêu cầu 1 R2; P2 giữ 2 R2 và yêu cầu 1 R1 và 1 R3; P3 giữ 1 R1 và yêu cầu 1 R2; P4 giữ 2 R3 và yêu cầu 1 R1.

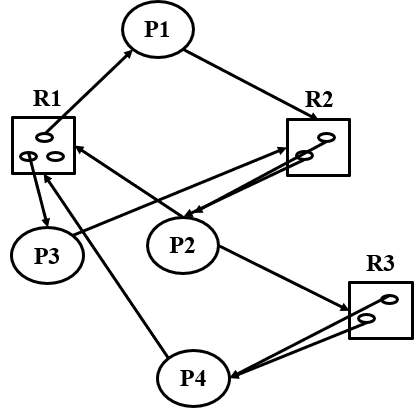
a. Vẽ đồ thị cấp phát tài nguyên của hệ thống

b. Hệ thống có deadlock không?

c. Tìm chuỗi an toàn (nếu có)

Trả lời:

1. Đồ thị cấp phát tài nguyên



1. Hệ thống không bị deadlock do hệ thống có chuỗi an toàn.
2. Chuỗi an toàn là: <P4, P2, P3, P1> hoặc <P4, P2, P1, P3>.
3. Cho 1 hệ thống có 5 tiến trình P1, P2, P3, P4, P5 và 3 loại tài nguyên R1 (có 3 thực thể), R2 (có 3 thực thể) R3 (có 2 thực thể). P1 giữ 1 thực thể R1 và yêu cầu 1 thực thể R2; P2 giữ 2 thực thể R2 và yêu cầu 1 thực thể R1 và 1 thực thể R3; P3 giữ 1 thực thể R1 và yêu cầu 1 thực thể R2; P4 giữ 2 thực thể R3 và yêu cầu 1 thực thể R1; P5 đang giữ 1 thực thể của R2 và yêu cầu 1 thự thể của R1.
4. Vẽ đồ thị cấp phát tài nguyên
5. Có bao nhiêu chuỗi an toàn cho hệ thống trên?
6. Viết ra tất cả các chuỗi an toàn (nếu có)
7. (Bài tập mẫu) Xét một hệ thống máy tính có 5 tiến trình: P0, P1, P2, P3, P4 và 4 loại tài nguyên: A, B, C, D. Tại thời điểm t0, trạng thái của hệ thống như sau:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | |
| **Tiến trình** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 |
| **P1** | 1 | 0 | 0 | 0 | 1 | 7 | 5 | 0 |
| **P2** | 1 | 3 | 5 | 4 | 2 | 3 | 5 | 6 |
| **P3** | 0 | 6 | 3 | 2 | 0 | 6 | 5 | 2 |
| **P4** | 0 | 0 | 1 | 4 | 0 | 6 | 5 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 1 | 5 | 2 | 0 |

a. Tìm Need?

b. Hệ thống có an toàn không?

c. Nếu P1 yêu cầu (0,4,2,0) thì có thể cấp phát cho nó ngay không?

Trả lời:

1. Ma trận Need

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Need** | | | |
| **Tiến trình** | **A** | **B** | **C** | **D** |
| **P0** | 0 | 0 | 0 | 0 |
| **P1** | 0 | 7 | 5 | 0 |
| **P2** | 1 | 0 | 0 | 2 |
| **P3** | 0 | 0 | 2 | 0 |
| **P4** | 0 | 6 | 4 | 2 |

1. Thực hiện giải thuật an toàn

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Allocation | | | | Max | | | | Need | | | | Available (Work) | | | |  |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |  |
| **P0** | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 5 | 2 | 0 | P0 |
| **P1** | 1 | 0 | 0 | 0 | 1 | 7 | 5 | 0 | 0 | 7 | 5 | 0 | 1 | 5 | 3 | 2 | P2 |
| **P2** | 1 | 3 | 5 | 4 | 2 | 3 | 5 | 6 | 1 | 0 | 0 | 2 | 2 | 8 | 8 | 6 | P3 |
| **P3** | 0 | 6 | 3 | 2 | 0 | 6 | 5 | 2 | 0 | 0 | 2 | 0 | 2 | 14 | 11 | 8 | P4 |
| **P4** | 0 | 0 | 1 | 4 | 0 | 6 | 5 | 6 | 0 | 6 | 4 | 2 | 2 | 14 | 12 | 12 | P1 |

Hệ thống có chuỗi an toàn <P0, P2, P3, P4, P1> cho nên hệ thống an toàn.

Request P1 (0,4,2,0) ≤ Need P1 (0, 7, 5, 0).

Request P1 (0,4,2,0) ≤ Available (1, 5, 2, 0).

Giả sử hệ thống đáp ứng yêu cầu (0,4,2,0) của P1.

Trạng thái mới của hệ thống:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Allocation | | | | Max | | | | Need | | | | Available (Work) | | | |  |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |  |
| **P0** | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | P0 |
| **P1** | 1 | 4 | 2 | 0 | 1 | 7 | 5 | 0 | 0 | 3 | 3 | 0 | 1 | 1 | 1 | 2 | P2 |
| **P2** | 1 | 3 | 5 | 4 | 2 | 3 | 5 | 6 | 1 | 0 | 0 | 2 | 2 | 4 | 6 | 6 | P3 |
| **P3** | 0 | 6 | 3 | 2 | 0 | 6 | 5 | 2 | 0 | 0 | 2 | 0 | 2 | 10 | 9 | 8 | P4 |
| **P4** | 0 | 0 | 1 | 4 | 0 | 6 | 5 | 6 | 0 | 6 | 4 | 2 | 2 | 10 | 7 | 12 | P1 |

Hệ thống mới vẫn có chuỗi an toàn <P0, P2, P3, P4, P1> cho nên hệ thống đáp ứng yêu cầu cấp phát cho P1.

1. Xét hệ thống tại thời điểm t0 có 5 tiến trình: P1, P2, P3, P4, P5; và 4 loại tài nguyên: R1, R2, R3, R4. Xét trạng thái hệ thống như sau:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Allocation | | | | Max | | | |
| Process | R1 | R2 | R3 | R4 | R1 | R2 | R3 | R4 |
| P1 | 0 | 0 | 1 | 2 | 0 | 0 | 3 | 2 |
| P2 | 2 | 0 | 0 | 0 | 2 | 7 | 5 | 0 |
| P3 | 0 | 0 | 3 | 4 | 6 | 6 | 5 | 6 |
| P4 | 2 | 3 | 5 | 4 | 3 | 3 | 5 | 6 |
| P5 | 0 | 3 | 3 | 2 | 0 | 6 | 5 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  |  |  |
|  |  |  |  |

1. Tại thời điểm t0, áp dụng giải thuật banker tìm chuỗi an toàn của hệ thống?
2. Tại thời điểm t1, tiến trình P3 yêu cầu thêm tài nguyên (1, 1, 0, 0) thì hệ thống có thể đáp ứng ngay được không? Tại sao?
3. Sử dụng giải thuật Banker để kiểm tra các trạng thái sau có an toàn hay không? Nếu có thì đưa ra chuỗi thực thi an toàn, nếu không thì nêu rõ lý do không an toàn?

a. Available = (0,3,0,1)

b. Available = (1,0,0,2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | |
| **Tiến trình** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 3 | 0 | 1 | 4 | 5 | 1 | 1 | 7 |
| **P1** | 2 | 2 | 1 | 0 | 3 | 2 | 1 | 1 |
| **P2** | 3 | 1 | 2 | 1 | 3 | 3 | 2 | 1 |
| **P3** | 0 | 5 | 1 | 0 | 4 | 6 | 1 | 2 |
| **P4** | 4 | 2 | 1 | 2 | 6 | 3 | 2 | 5 |

1. Trả lời các câu hỏi sau bằng cách sử dụng giải thuật Banker:

a. Hệ thống có an toàn không? Đưa ra chuỗi an toàn nếu có?

b. Nếu P1 yêu cầu (1,1,0,0) thì có thể cấp phát cho nó ngay không?

c. Nếu P4 yêu cầu (0,0,2,0) thì có thể cấp phát cho nó ngay không?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | |
| **Tiến trình** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |