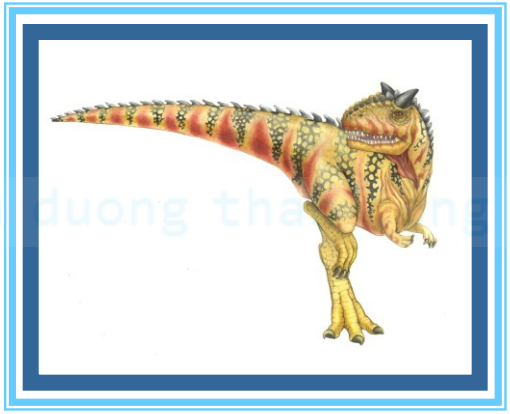
Chương 6: Deadlocks



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Câu hỏi ôn tập chương 5

- Phân biệt semaphore với monitor? Nêu ứng dụng của từng giải pháp?
- Áp dụng semaphore vào bài toán reader-writer, giải thích rõ hoạt động?

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Mục tiêu chương 6

- Hiểu được vấn đề bài toán deadlock và các tính chất của deadlock
- Hiển được các phương pháp giải quyết deadlock
  - ~~Bảo vệ~~ Ngăn (Deadlock prevention)
  - Tránh (Deadlock avoidance)
  - Kiểm tra (Deadlock detection)
  - Phục hồi (Deadlock recovery)

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Nội dung

- Bài toán deadlock
- Mô hình hệ thống
- Các tính chất của deadlock
- Phương pháp giải quyết deadlock

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Vấn đề deadlock

- *Tình huống:* Một tập các tiến trình bị block, mỗi tiến trình giữ tài nguyên và đang chờ tài nguyên mà tiến trình khác trong tập đang giữ
- Ví dụ 1:
  - Hệ thống có 2 file A và B trên đĩa
  - P1 và P2 mỗi tiến trình mở một file và yêu cầu mở file kia.

  P1 đã mở, đang nắm giữ file A và yêu cầu file B; trong khi P2 đã mở, đang nắm giữ file B và yêu cầu mở file A.

  P1 muốn hoàn tất thì phải có cả file A và B, P2 cũng vậy.
- Ví dụ 2:
  - Bài toán các triết gia ăn tối
  - Mỗi người cầm 1 chiếc đũa và chờ chiếc còn lại

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Mô hình hóa hệ thống

- Các loại tài nguyên, kí hiệu R1, R2,…,Rm, bao gồm:
  - CPU cycle, không gian bộ nhớ, thiết bị I/O, file, semaphore,..
  - Mỗi loại tài nguyên Ri có Wi thực thể
- Giả sử tài nguyên tái sử dụng theo chu kỳ
  - Yêu cầu: tiến trình phải chờ nếu yêu cầu không được đáp ứng ngày
  - Sử dụng: tiến trình sử dụng tài nguyên
  - Hoàn trả: tiến trình hoàn trả tài nguyên
- Các tác vụ yêu cầu và hoàn trả đều là system call. Ví dụ:
  - Request/release device
  - Open/close file
  - Allocate/free memory
  - Wait/signal

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Định nghĩa

- Một tiến trình gọi là *deadlock* nếu nó đang đợi một sự kiện mà sẽ không bao giờ xảy ra
  - Thông thường, có nhiều hơn một tiến trình bị liên quan trong một deadlock
- 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
  - Ví dụ: Một tiến trình sẵn sàng để xử lý nhưng nó không bao giờ nhận được CPU

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Điều kiện cần để xảy ra deadlock

- *Loại trừ hỗ tương (Mutual exclusion):* ít nhất một tài nguyên được giữ theo nonsharable mode
- *Giữ và chờ cấp thêm tài nguyên (Hold and wait)*: Một tiến trình đang giữ ít nhất một tài nguyên và đợi thêm tài nguyên do quá trình khác giữ

Điều kiện cần để xảy ra deadlock (tt)

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- *Không trưng dụng (No Preemption):* 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 vòng tròn (Circular wait)*: tồn tại một tập (P0,…,Pn} các quá trình đang đợi sao cho
  - P_0 đợi một tài nguyên mà P_1 giữ
  - P_1 đợi một tài nguyên mà P_2 giữ
  - …
  - P_n đợi một tài nguyên mà P_0 giữ

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

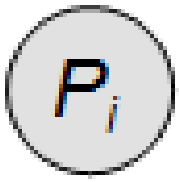
Đồ thị cấp phát tài nguyên - RAG

RAG: Resource-Allocation Graph

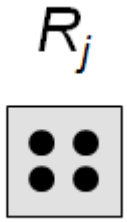
- Là đồ thị có hướng, với tập đỉnh V và tập cạnh E
- Tập đỉnh V gồm 2 loại:
  - $P = \{P_1, P_2, \ldots, P_n\}$ (All process)
  - $R = \{R_1, R_2, \ldots, R_n\}$ (All resource)
- Tập cạnh E gồm 2 loại:
  - Cạnh yêu cầu: $P_i \rightarrow R_j$
  - Cạnh cấp phát: $R_j \rightarrow P_i$

Đồ thị cấp phát tài nguyên – RAG (tt)

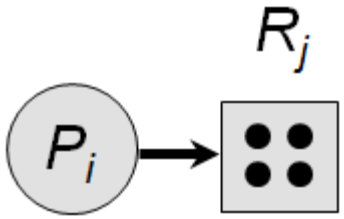
- Process i



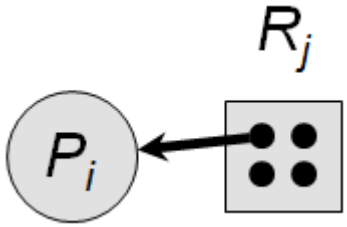
- Loại tài nguyên R_j với 4 thực thể



- P_i yêu cầu một thực thể của R_j

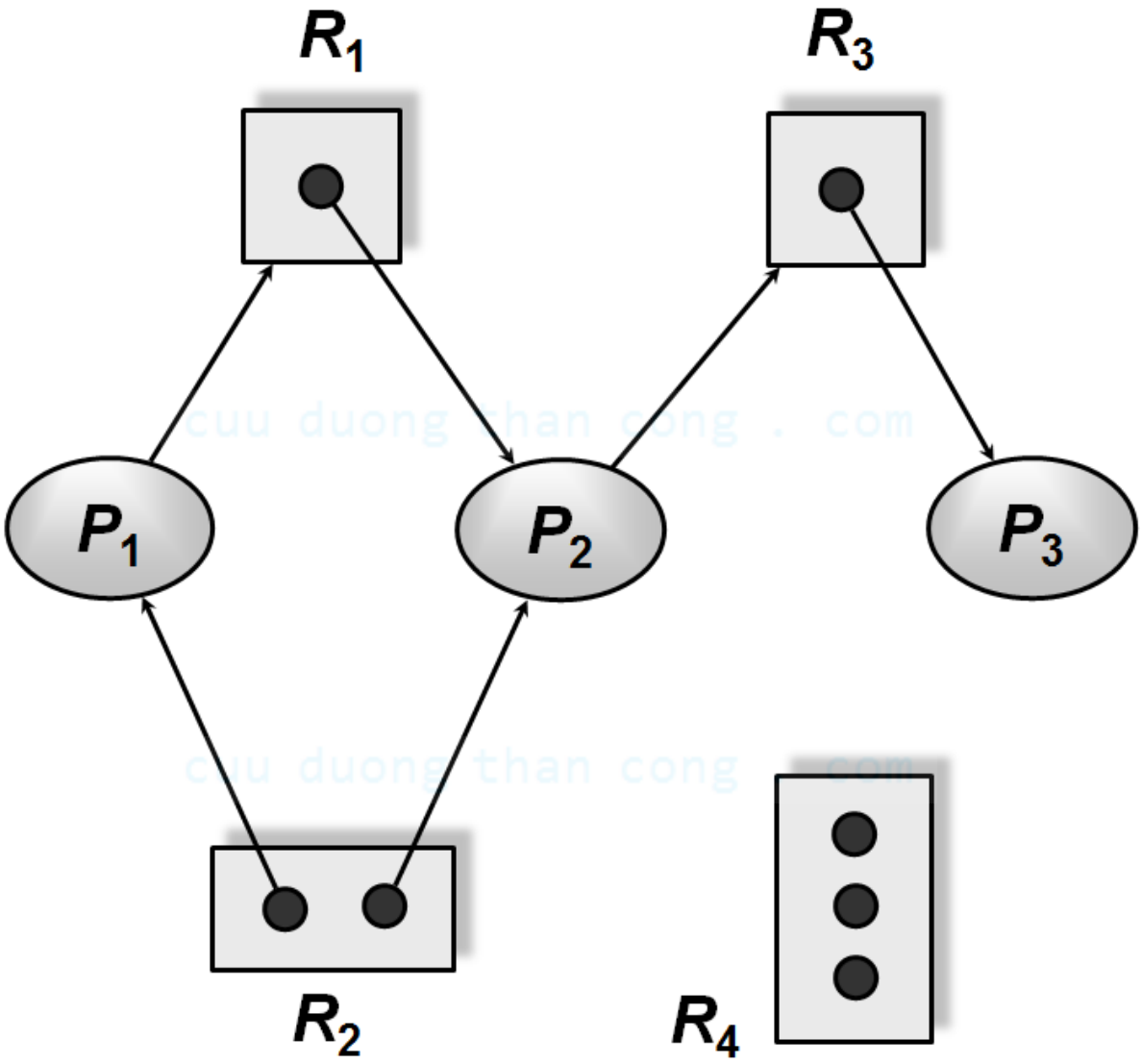


- P_i đang giữ một thực thể của R_j

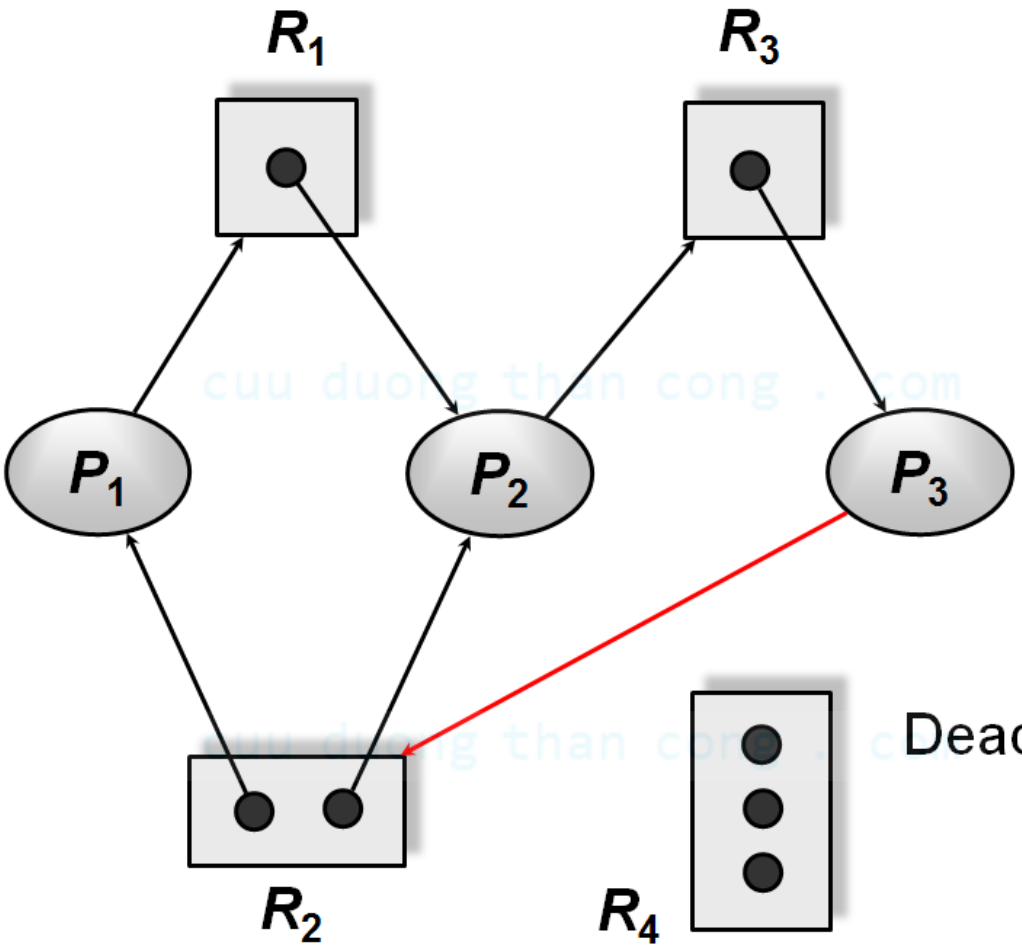


UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ví dụ RAG

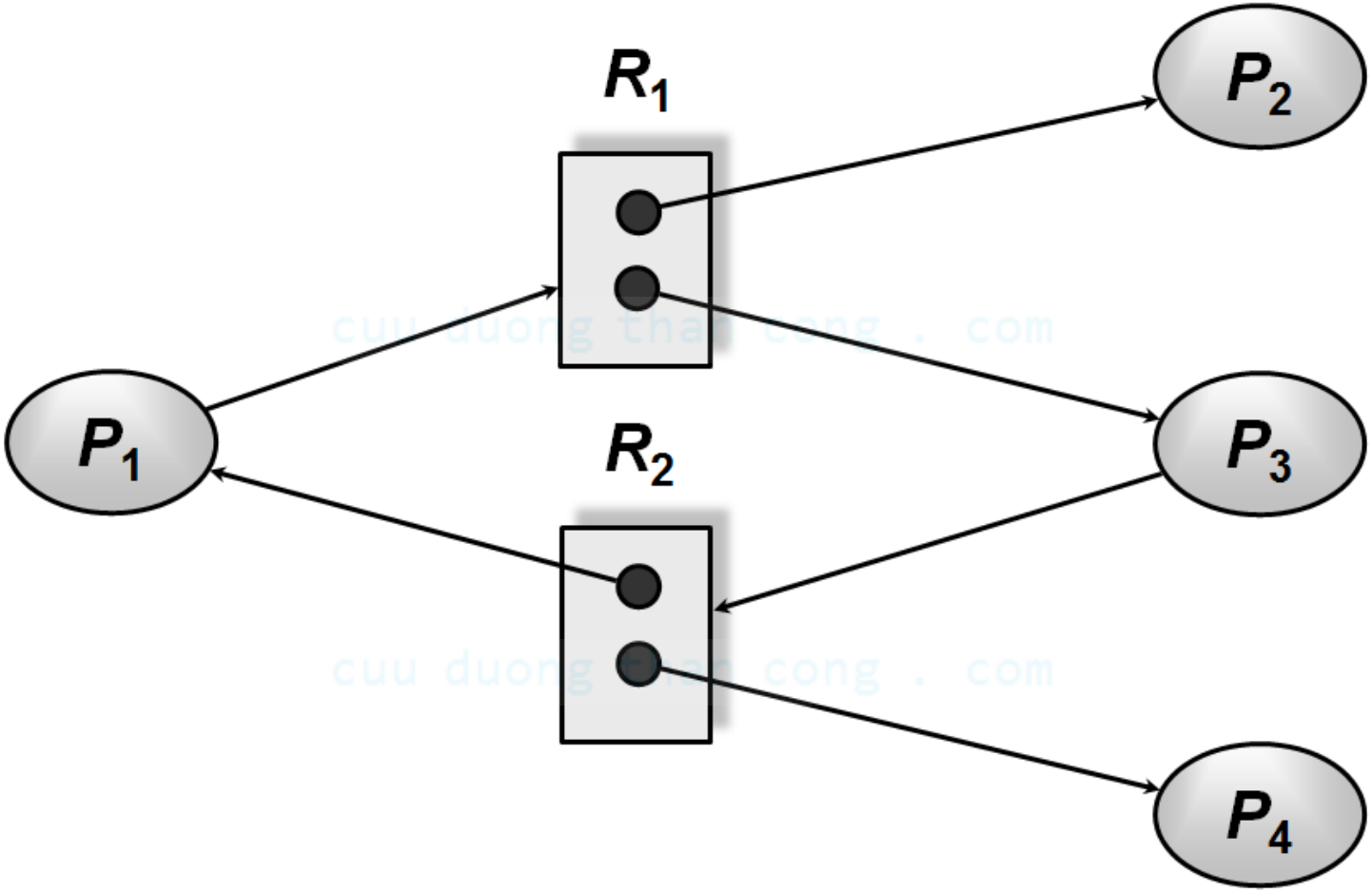


Đồ thị cấp phát tài nguyên với một deadlock



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Đồ thị chứa chu trình nhưng không deadlock



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

RAG và deadlock

- RAG không chứa chu trình → không có deadlock
- RAG chứa một (hay nhiều) chu trình
  - Nếu mỗi loại tài nguyên chỉ có một thực thể

    → deadlock
  - Nếu mỗi loại tài nguyên có nhiều thực thể

    → có thể xảy ra deadlock

Các phương pháp giải quyết deadlock

- Bảo đảm rằng hệ thống không rơi vào tình trạng deadlock bằng cách *ngăn* (Prevention) hoặc *tránh* (Avoidance) deadlock
- Khác biệt
  - *Ngăn deadlock*: không cho phép (ít nhất) một trong 4 điều kiện cần cho deadlock
  - *Tránh deadlock*: các quá trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp

Các phương pháp giải quyết deadlock (tt)

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock (Detection) và phục hồi hệ thống (Recovery)
- Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống
  - Khá nhiều hệ điều hành sử dụng phương pháp này
  - 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ngăn deadlock

- Ngăn deadlock bằng cách ngăn một trong 4 điều kiện cần của deadlock
- Ngăn mutual exclusion
  - Đối với tài nguyên không chia sẻ (ví dụ: printer): không làm được
  - Đối với tài nguyên chia sẻ (ví dụ: read-only file): không cần thiết

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ngăn deadlock (tt)

- Hold and wait
  - Cách 1: Mỗi tiến trình yêu cầu toàn bộ tài nguyên cần thiết một lần. Nếu có đủ tài nguyên thì hệ thống sẽ cấp phát, nếu không đủ tài nguyên thì tiến trình phải bị block
  - Cách 2: Khi yêu cầu tài nguyên, tiến trình không được giữ tài nguyên nào. Nếu đang có thì phải trả lại trước khi yêu cầu

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ngăn deadlock (tt)

- Ngăn no preemption: nếu tiến trình A có giữ tài nguyên và đang yêu cầu tài nguyên khác nhưng tài nguyên này chưa được cấp phát ngay thì:
  - Cách 1: Hệ thống lấy lại mọi tài nguyên mà A đang giữ
    - A chỉ bắt đầu lại được khi có được các tài nguyên đã bị lấy lại cùng với tài nguyên đang yêu cầu
  - Cách 2: Hệ thống sẽ xem tài nguyên mà A yêu cầu
    - Nếu tài nguyên được giữ bởi một tiến trình khác đang đợi thêm tài nguyên, tài nguyên này được hệ thống lấy lại và cấp phát cho A
    - Nếu tài nguyên được giữ bởi tiến trình không đợi tài nguyên, A phải đợi và tài nguyên của A bị lấy lại. Tuy nhiên hệ thống chỉ lấy lại các tài nguyên mà tiến trình khác yêu cầu

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ngăn deadlock (tt)

- Ngăn Circular wait: gán một thứ tự cho tất cả các tài nguyên trong hệ thống
  - Tập hợp tài nguyên: $R = \{R_1, R_2, \ldots, R_m\}$
    - Hàm ánh xạ: F: R → N (Với N là tập hợp các số tự nhiên)

Ví dụ: Có 3 tài nguyên *tape drive*, *disk drive* và *printer*. Và hàm F ví dụ như sau:

F(*tape drive*) = 1

F (*disk drive*) = 5

F (*printer*) = 12

Ngăn deadlock (tt)

- Ngăn Circular wait (tt):

Dựa vào hàm F như trên, một cách như sau có thể được sử dụng để ngăn Circular wait:

Mỗi tiến trình có thể yêu cầu thực thể của một loại tài nguyên chỉ theo thứ tự tăng dần (định nghĩa bởi hàm F).

Cụ thể, một process đầu tiên yêu cầu thực thể R_i nào đó. Sau đó, process này chỉ có thể yêu cầu thêm thực thể R_j nếu và chỉ nếu $F(R_j) > F(R_i)$

(Tất nhiên, người lập trình có thể chọn cách khác, ví dụ process sau khi đã có Ri, chỉ có thể yêu cầu Rj nếu và chỉ nếu $F(R_j) \leq F(R_i)$

Ngăn deadlock (tt)

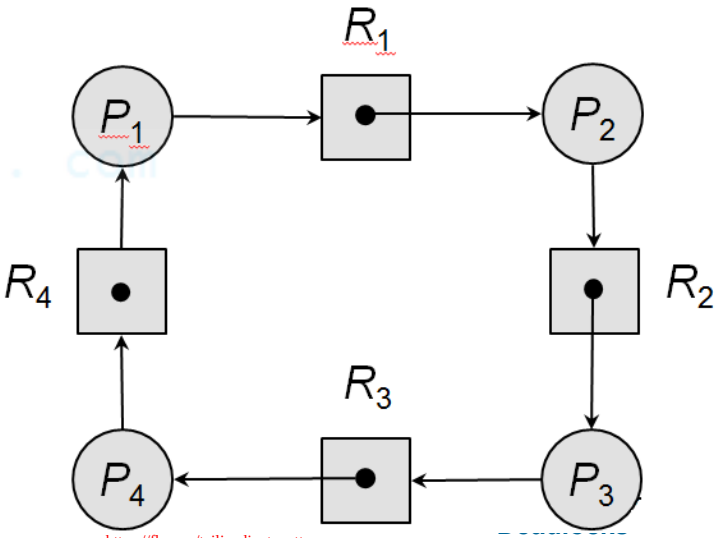
UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ngăn Circular wait (tt):

Chứng minh cơ chế này có thể ngăn Circular wait bằng phản chứng:

- Giả sử tồn tại 4 process (P_1, P_2, P_3, P_4) với 4 thực thể tài nguyên (R_1, R_2, R_3, R_4) tạo thành một chu trình bị deadlock như hình.
- Một process P_i sẽ giữ thực thể tài nguyên R_{i-1} (Với P_0 thì P_0 giữ thực thể tài nguyên R_4) và đang yêu cầu thực thể tài nguyên R_i
- Giả sử yêu cầu tài nguyên từ P_1 tới P_4 đều thỏa (sẽ tạo ra deadlock), theo quy ước của cơ chế trên, phải có:
  - F(R4) < F(R1)
  - F(R1) < F(R2)
  - F(R2) < F(R3)
  - F(R3) < F(R4)

➔ Vậy F(R4) < F(R4) ➔ mâu thuẫn



Ngăn deadlock (tt)

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Ngăn Circular wait (tt):

Lưu ý rằng: *Việc tạo ra hàm F và cấp phát theo thứ tự tự bản thân nó không thể ngăn deadlock, mà phụ thuộc vào cách mà người lập trình lập trình như thế nào.*

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Tránh deadlock

- Ngăn deadlock sử dụng tài nguyên không hiệu quả
- Tránh deadlock vẫn đảm bảo hiệu suất sử dụng tài nguyên tối đa đến mức có thể
- Yêu cầu mỗi tiến trình khai báo số lượng tài nguyên tối đa cần để thực hiện công việc
- Giải thuật tránh deadlock sẽ kiểm tra trạng thái cấp phát tài nguyên để đảm bảo hệ thống không rơi vào deadlock
- Trạng thái cấp phát tài nguyên được định nghĩa dựa trên số tài nguyên còn lại, số tài nguyên đã được cấp phát và yêu cầu tối đa của các tiến trình

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Trạng thái safe và unsafe

- Một trạng thái của hệ thống được gọi là an toàn (safe) nếu tồn tại một chuỗi thứ tự an toàn
- Một chuỗi quá trình $<P_1, P_2, \ldots, P_n>$ là một chuỗi an toàn nếu
  - Với mọi i = 1, …, n yêu cầu tối đa về tài nguyên của P_i có thể được thỏa bởi:
    - Tài nguyên mà hệ thống đang có sẵn sàng
    - Cùng với tài nguyên mà tất cả các P_j (j<i) đang giữ
- Một trạng thái của hệ thống được gọi là không an toàn (unsafe) nếu không tồn tại một chuỗi an toàn

Trạng thái safe và unsafe (tt)

- Ví dụ: hệ thống có 12 tape drive và 3 tiến trình P_0, P_1, P_2
  - Tại thời điểm t_o

	Cần tối đa	Đang giữ	Cần thêm
P_0	10	5	5
P_1	4	2	2
P_2	9	2	7

  - Còn 3 tape drive sẵn sàng
  - Chuỗi **<P_1, P_0, P_2>** là chuỗi an toàn → hệ thống là an toàn

Trạng thái safe và unsafe (tt)

- Giả sử tại thời điểm t_1, P_2 yêu cầu và được cấp phát 1 tape drive
  - Còn 2 tape drive sẵn sàng

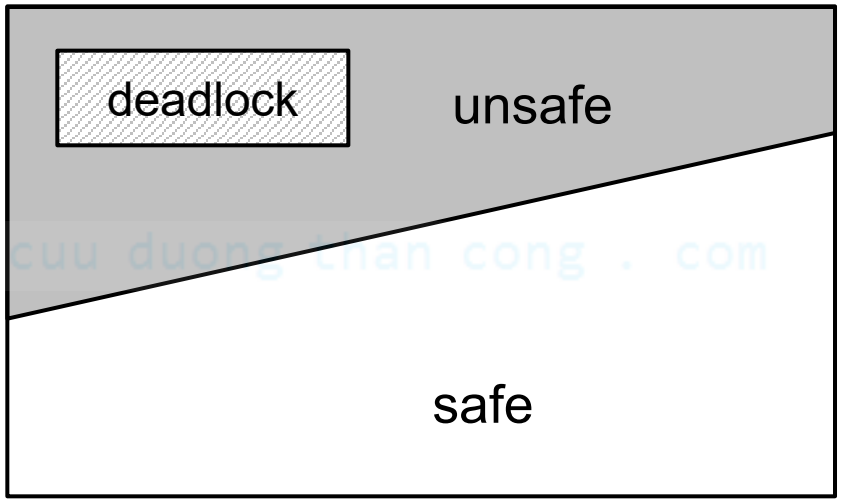
	Cần tối đa	Đang giữ
P_0	10	5
P_1	4	2
P_2	9	3

  - Hệ thống còn an toàn không? ➔ Không

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Trạng thái safe/unsafe và deadlock

- Nếu hệ thống đang ở trạng thái safe → không deadlock
- Nếu hệ thống đang ở trạng thái unsafe → có thể dẫn đến deadlock
- Tránh deadlock bằng cách bảo đảm hệ thống không đi đến trạng thái unsafe



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Ôn tập

- Khái niệm deadlock
- Các tính chất của deadlock
- Đồ thị cấp phát tài nguyên
- Các phương pháp giải quyết deadlock
- Ngăn deadlock
- Tránh deadlock