BÀI 2

QUẢN LÝ TIẾN TRÌNH

TÌNH HUỐNG DẪN NHẬP

- Các chương trình trong máy tính hoạt động như thế nào?
- Tạo sao cùng một lúc ta có thể vừa chạy được đồng hồ ở máy tính, vừa chat được?
- Nhiều khi máy tính rơi vào trạng thái treo. Tại sao lại có hiện tượng đó?

MỤC TIÊU

- Nắm được khái niệm về tiến trình, các trạng thái của tiến trình và quá trình biến đổi trạng thái đó.
- Nắm được các khái niệm về luồng, đồng bộ và giải quyết tranh chấp.
- Nắm được những vấn đề liên quan đến deadlock.

NỘI DUNG

1. Khái niệm tiến trình;
2. Đồng bộ và giải quyết tranh chấp;
3. Tắc nghẽn (Deadlock).

1. KHÁI NIỆM TIẾN TRÌNH

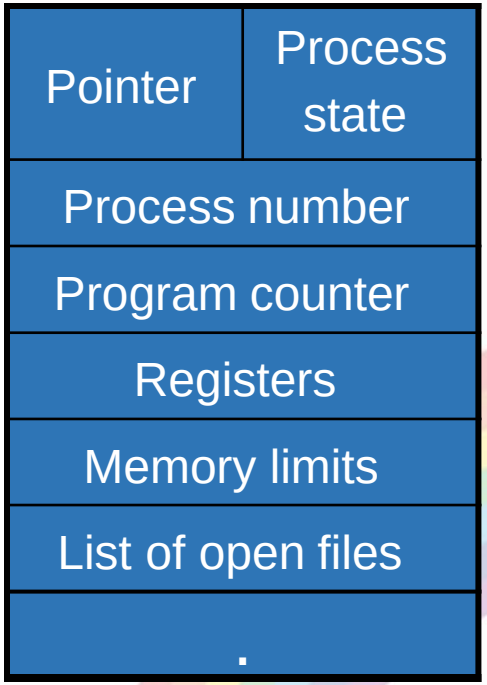
1. KHÁI NIỆM TIẾN TRÌNH

Thường dùng định nghĩa tiến trình như là một chương trình trong lúc chạy.

1. KHÁI NIỆM TIẾN TRÌNH

1.1. KHỐI ĐIỀU KHIỂN TIẾN TRÌNH PCB

- PCB – Process Control Block:
  - Là một cấu trúc dữ liệu;
  - Chứa thông tin quan trọng về tiến trình;
  - Có thể khác nhau trong các hệ thống khác nhau.
- PCB là đối tượng chính đại diện cho tiến trình đối với hệ điều hành.
- Cấu trúc của một PCB (hình bên):
- Hệ điều hành nhờ PCB có thể có những thông tin cơ bản về một tiến trình:
  - Trạng thái của tiến trình;
  - ID (Identifier) duy nhất cho tiến trình;
  - Độ ưu tiên (Priority) của tiến trình;
  - Thông tin về bộ nhớ;
  - Thông tin về các tài nguyên tiến trình đang sử dụng;
  - Vùng để cho các thanh ghi.



1. KHÁI NIỆM TIẾN TRÌNH

1.2. TRẠNG THÁI CỦA TIẾN TRÌNH

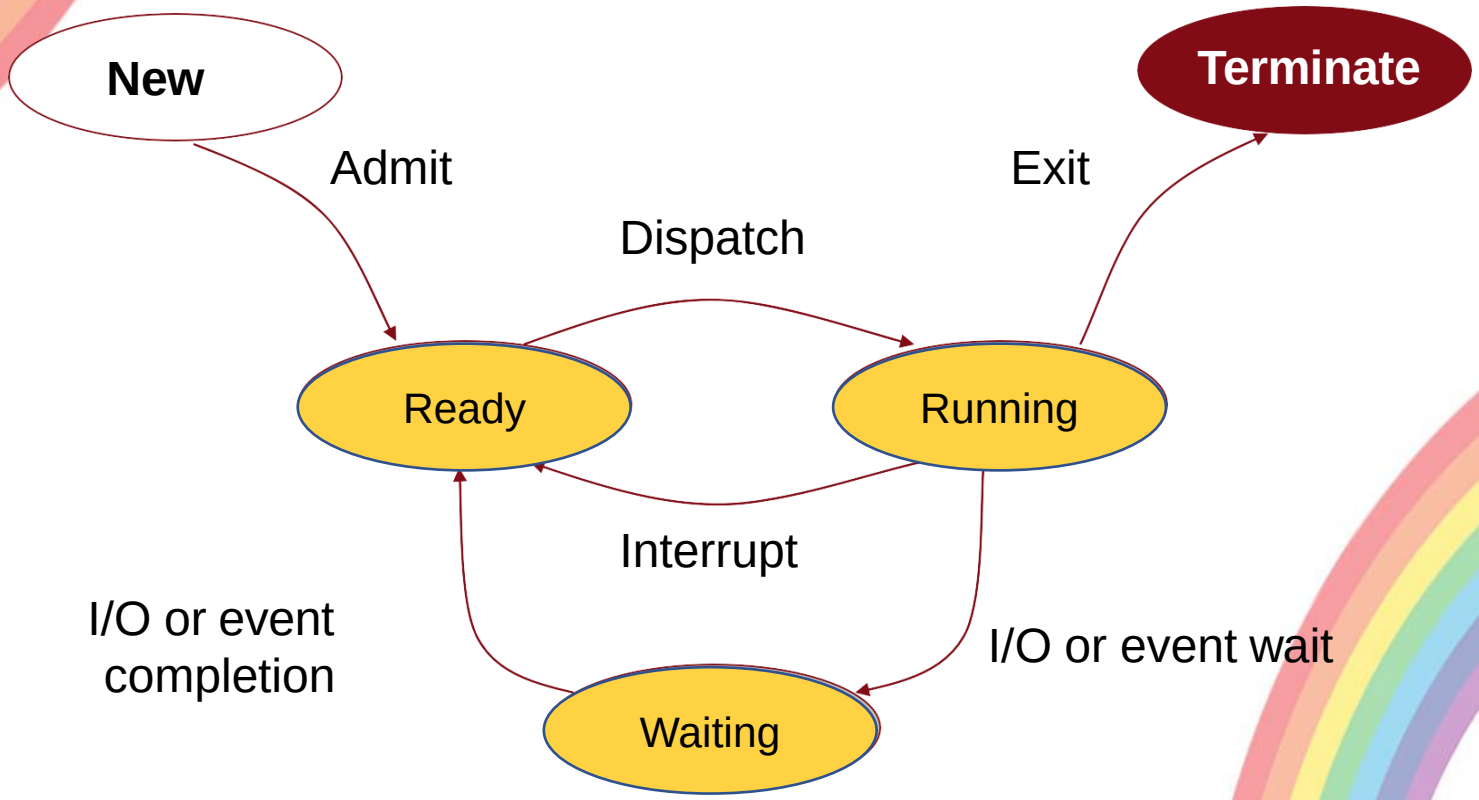
- Khởi tạo (New): Tiến trình vừa được tạo;
- Sẵn sàng (Ready): Tiến trình đã có đủ tài nguyên, chỉ còn cần CPU;
- Thực hiện (Running): Các lệnh của tiến trình đang được thực thi;
- Chờ/ dừng (Waiting/ Blocked): Tiến trình đợi I/O hoàn tất, tín hiệu;
- Kết thúc (Terminated): Tiến trình đã kết thúc.

1. KHÁI NIỆM TIẾN TRÌNH

1.2. TRẠNG THÁI CỦA TIẾN TRÌNH

BIẾN ĐỔI TRẠNG THÁI CỦA TIẾN TRÌNH

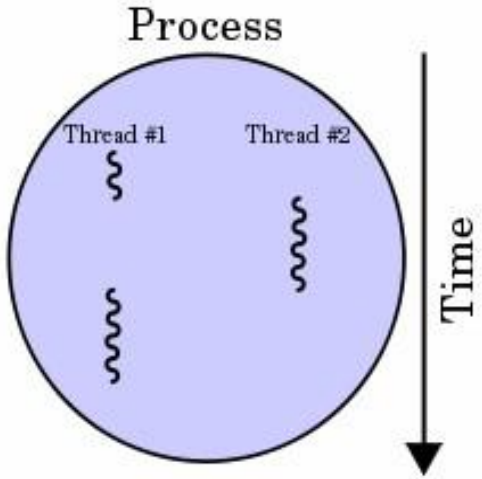
Chuyển đổi trạng thái giữa các tiến trình



1. KHÁI NIỆM TIẾN TRÌNH

1.3. LUỒNG

- Khái niệm: Luồng là một dòng điều khiển trong phạm vi một quá trình;
- Tiến trình đa luồng gồm nhiều dòng điều khiển khác nhau trong cùng không gian địa chỉ và được gọi là tiến trình nhẹ (Lightweight Process-LWP);
- Luồng là một đơn vị cơ bản của việc sử dụng CPU;
- Nó hình thành gồm:
  - Một định danh luồng (Thread ID);
  - Một bộ đếm chương trình;
  - Tập thanh ghi và ngăn xếp.
- Nếu tiến trình có nhiều luồng điều khiển, nó có thể thực hiện nhiều hơn một tác vụ tại một thời điểm.

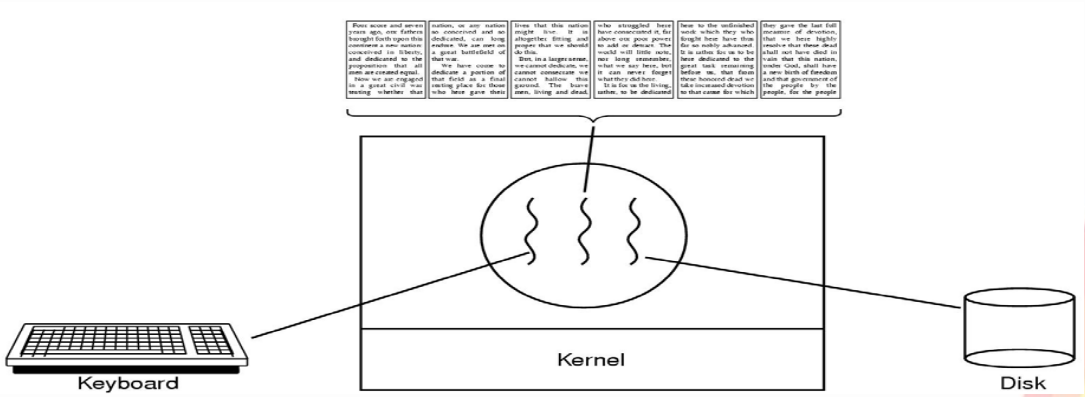


Tiến trình đa luồng với hệ thống 1 bộ xử lý

1. KHÁI NIỆM TIẾN TRÌNH

1.4. ĐA LUỒNG

- Quá trình đa luồng (Multithreaded Process) là tiến trình có nhiều luồng.
  Ví dụ: A Word Processor With Three Threads.
- Ưu điểm:
  - Tính đáp ứng (Responsiveness);
  - Chia sẻ tài nguyên (Resource sharing);
  - Tiết kiệm chi phí hệ thống (Economy);
  - Chi phí tạo/quản lý Thread nhỏ hơn so với quá trình;
  - Chi phí chuyển ngữ cảnh giữa các Thread nhỏ hơn so với tiến trình;
  - Tận dụng kiến trúc đa xử lý (Multiprocessor).



1. KHÁI NIỆM TIẾN TRÌNH

1.5. XỬ LÝ NGẮT

- Ngắt (Interupt) là sự kiện làm thay đổi trình tự thực hiện lệnh bình thường của bộ xử lý;
- Tín hiệu ngắt được xử lý bởi phần cứng;
- Trình tự thực hiện khi xử lý ngắt:
  - Điều khiển chuyển cho hệ điều hành;
  - Hệ điều hành lưu lại trạng thái của tiến trình bị ngắt;
  - Hệ điều hành phân tích loại ngắt và chuyển điều khiển cho chương trình xử lý ngắt tương ứng.
- Các dạng ngắt:
  - SVC- Interrupt;
  - Ngắt vào/ra;
  - External Interrupt;
  - Restart Interrupt;
  - Program Check Interrupt;
  - Machine Check Interrupt.

1. KHÁI NIỆM TIẾN TRÌNH

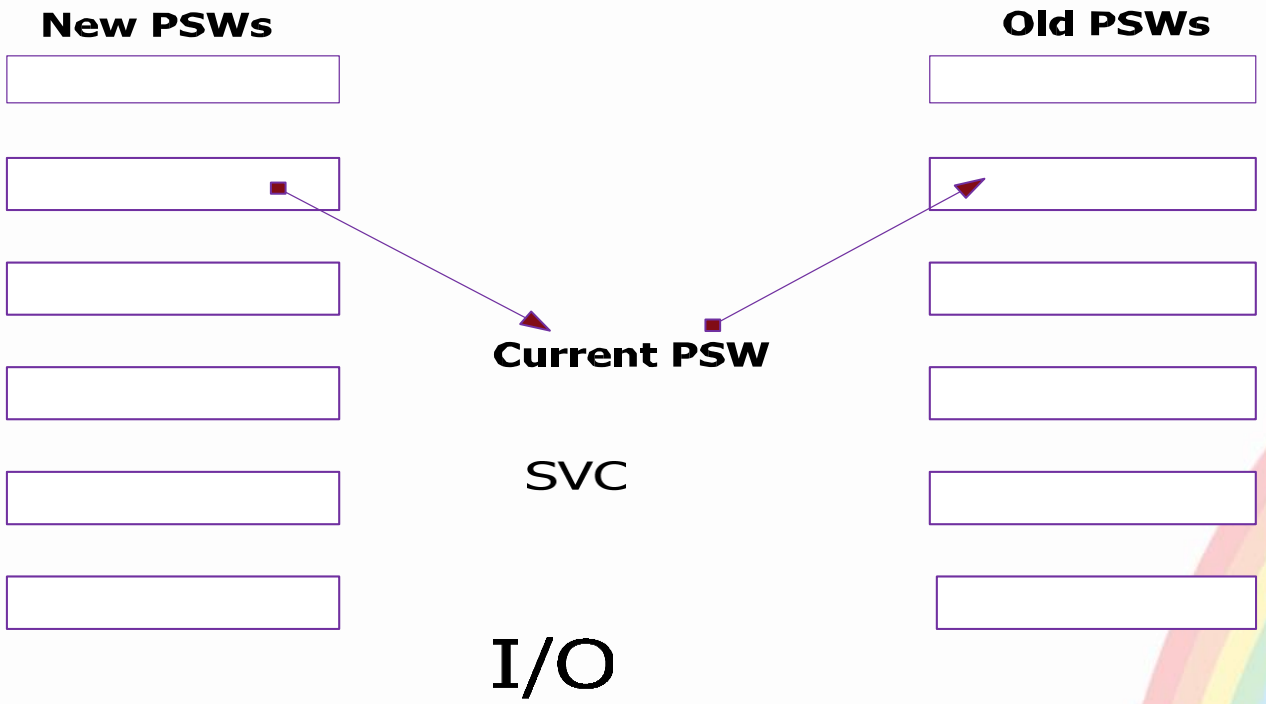
1.6. ĐỔI NGỮ CẢNH

- Ngữ cảnh (Context) của một tiến trình là trạng thái của tiến trình;
- Ngữ cảnh của tiến trình được biểu diễn trong PCB của nó;
- Đổi ngữ cảnh (Context switch) là công việc giao CPU cho tiến trình khác. Khi đó cần:
  - Lưu ngữ cảnh của tiến trình cũ vào PCB của nó;
  - Nạp ngữ cảnh từ PCB của tiến trình mới để tiến trình mới thực thi.
- Sử dụng các thanh ghi trạng thái chương trình PSW (Program Status Word);
- Có 3 loại PSW:
  - PSW hiện thời (Current);
  - PSW mới (New);
  - PSW cũ (Old).

1. KHÁI NIỆM TIẾN TRÌNH

1.6. ĐỔI NGỮ CẢNH

- Trong hệ có một bộ xử lý thì chỉ có 1 Current PSW, nhưng có 6 New PSW (tương ứng cho mỗi loại ngắt) và 6 Old PSW tương ứng.



1. KHÁI NIỆM TIẾN TRÌNH

1.7. THIẾT KẾ PHÂN LỚP

- Kiến trúc phân cấp của hệ thống;
- Hạt nhân của OS;
- Các chức năng chính của Kernel;
- Cho phép (Enable) và cấm (Diasable) ngắt;
- Thực hiện Kernel với mã đơn giản nhất của máy tính.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.1. Tiến trình đồng thời;

2.2. Xử lý song song;

2.3. Vùng tranh chấp.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.1. TIẾN TRÌNH ĐỒNG THỜI

- Các tiến trình gọi là đồng thời/tương tranh nếu các tiến trình đó tồn tại đồng thời;
- Các tiến trình tương tranh (Concurent Process) có thể hoạt động hoàn toàn độc lập với nhau hoặc song song không đồng bộ asynchronous.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.2. XỬ LÝ SONG SONG

- Các lệnh chỉ thị xử lý song song: Parbegin và Parend;
- Các chỉ thị này thường đi theo cặp:
  - Chỉ thị đầu tiên chỉ ra rằng: Bắt đầu từ sau lệnh đó, chương trình được tách thành một số dòng điều khiển (Thread Control) thực hiện song song.
  - Chỉ thị thứ hai chỉ ra rằng: Từ đó chương trình lại được xử lý tuần tự.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

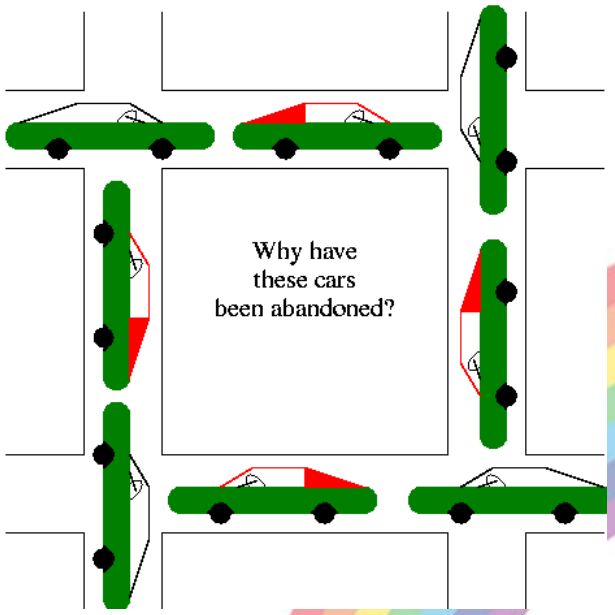
2.3. VÙNG TRANH CHẤP

- Khi có một tiến trình nằm trong khoảng tới hạn → Không cho phép tiến trình được vào khoảng tới hạn;
- Khi proces ra khỏi khoảng tới hạn → Một trong số các tiến trình đang chờ vào khoảng tới hạn phải được tiếp tục vào khoảng tới hạn;
- Khi tiến trình ra khỏi khoảng tới hạn → Hệ điều hành phải kiểm soát được để huỷ bỏ chế độ tới hạn.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.1. ĐỊNH NGHĨA DEADLOCK

- Trong hệ thống đa chương trình, một tiến trình nằm trong trạng thái Deadlock hay treo, nếu như nó chờ sự kiện (Event) nào đó không bao giờ xảy ra;
- Tình huống treo hệ thống là tình huống có một hay nhiều tiến trình nằm trong trạng thái treo.
- Ví dụ: Tình trạng tắc nghẽn giao thông.



2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.2. ĐIỀU KIỆN CẦN ĐỂ XẢY RA DEADLOCK

- Các tiến trình yêu cầu quyền độc quyền sử dụng tài nguyên sẽ cấp phát cho nó (điều kiện loại trừ nhau);
- Tiến trình giữ cho mình các tài nguyên đã được cấp và đồng thời yêu cầu tài nguyên bổ sung (điều kiện chờ tài nguyên);
- Tài nguyên không được lấy lại từ tiến trình khi các tài nguyên đó chưa được sử dụng để kết thúc công việc (điều kiện không phân chia);
- Tồn tại vòng kín các tiến trình, trong đó mỗi tiến trình giữ tài nguyên mà tiến trình kế tiếp đang đòi hỏi (điều kiện chờ vòng).

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.3. PHƯƠNG PHÁP GIẢI QUYẾT

Các nghiên cứu về Deadlock có thể chia ra làm 4 hướng sau:

- Ngăn chặn deadlock;
- Tránh deadlock;
- Phát hiện deadlock;
- Khôi phục sau deadlock.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.3. PHƯƠNG PHÁP GIẢI QUYẾT

NGĂN CHẶN DEADLOCK

- **Loại bỏ điều kiện "chờ tài nguyên bổ sung":**
  - 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 sẽ bị Blocked.
  - Cách 2: Khi yêu cầu tài nguyên, tiến trình không đang giữ bất kỳ tài nguyên nào. Nếu đang giữ thì phải trả lại trước khi yêu cầu.
- **Khuyết điểm:**
  - Làm tăng khả năng tiến trình phải chờ vô hạn vì vậy không kinh tế.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.3. PHƯƠNG PHÁP GIẢI QUYẾT

LOẠI BỎ ĐIỀU KIỆN "KHÔNG PHÂN BỐ LẠI"

- Nếu tiến trình A có giữ tài nguyên và đang yêu cầu tài nguyên khác nhưng tài nguyên này chưa cấp phát ngay được thì:
  - Cách 1:
    - ✓ Hệ thống lấy lại mọi tài nguyên mà A đang giữ;
    - ✓ A phải bắt đầu lại từ đầu.
  - 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.
- Khuyết điểm:
  - Có thể mất các kết quả làm;
  - Xác suất xảy ra 'chặn vô hạn' (Indefinite posponement);
  - Lãng phí tài nguyên;
  - Giảm hiệu suất của cả hệ thống.

2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP

2.3.3. PHƯƠNG PHÁP GIẢI QUYẾT

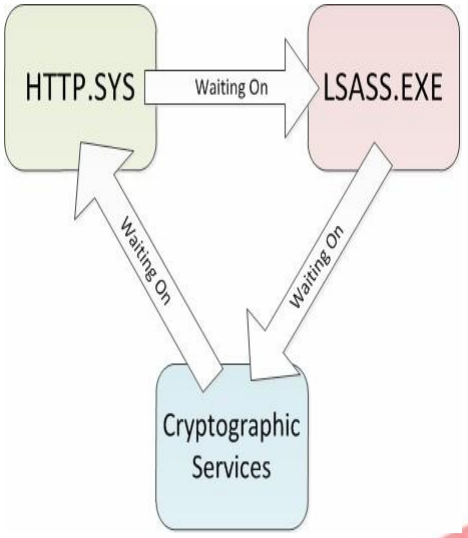
LOẠI TRỪ ĐIỀU KIỆN "CHỜ VÒNG QUANH"

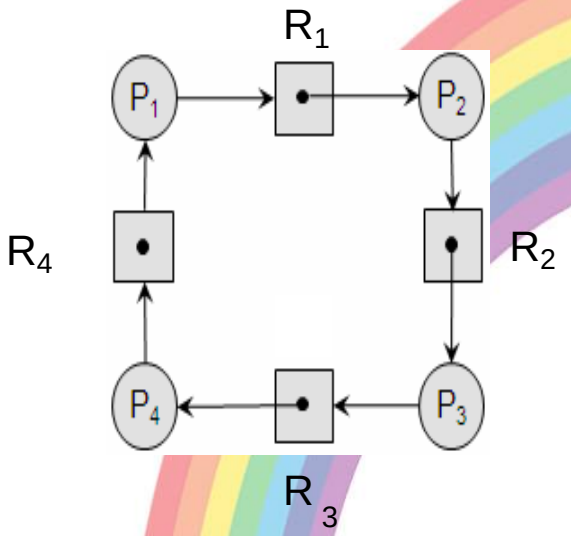
Tập các loại tài nguyên trong hệ thống được gán một thứ tự hoàn toàn.

- Cách 1: Mỗi tiến trình yêu cầu thực thể của tài nguyên theo thứ tự tăng dần (định nghĩa bởi hàm F) của loại tài nguyên.
- Cách 2: Khi một tiến trình yêu cầu một thực thể của loại tài nguyên R_j thì nó phải trả lại các tài nguyên R_i với $F(R_i) > F(R_j)$
- "Chứng minh" cho cách 1 bằng phương pháp phản chứng:
  - $F(R_4) < F(R_1)$;
  - $F(R_1) < F(R_2)$;
  - $F(R_2) < F(R_3)$;
  - $F(R_3) < F(R_4)$;

Vậy $F(R_4) < F(R_4)$, mâu thuẫn!

- Khuyết điểm: Lại ảnh hưởng xấu đến công việc của user trong tiến trình làm việc.





2. ĐỒNG BỘ VÀ GIẢI QUYẾT TRANH CHẤP

2.3. VÙNG TRANH CHẤP
2.3.3. PHƯƠNG PHÁP GIẢI QUYẾT

TRẠNG THÁI ỔN ĐỊNH VÀ KHÔNG ỔN ĐỊNH

- Trạng thái hiện thời của máy tính gọi là ổn định nếu hệ điều hành có thể đảm bảo tất cả các chương trình ứng dụng hiện thời trong hệ thống có thể hoàn thành sau một khoảng thời gian hữu hạn nào đó.
- Trạng thái ngược lại gọi là trạng thái không ổn định.

2. THUẬT TOÁN BANKER

- Giả sử rằng có n người sử dụng;
- Giả sử l(i) là số thiết bị cấp cho người sử dụng thứ i;
- Giả sử m(i) là số thiết bị lớn nhất mà người dùng thứ i có thể cần.
  - Tại một thời điểm, c(i) là yêu cầu lớn nhất hiện thời của người dùng I, bằng hiệu giữa số thiết bị nhiều nhất có thể yêu cầu và số thiết bị hiện có, tức là c(i) = m(i) - l(i);
  - Trong hệ thống với t thiết bị thì số thiết bị còn rỗi tại một thời điểm là a sẽ bằng t trừ tổng các thiết bị được cấp phát: a = t - ∑l(i).

2. THUẬT TOÁN BANKER

VÍ DỤ

Giả sử hệ thống có 12 thiết bị, và chúng được phân chia giữa 3 người dùng với trạng thái Status1 được biểu diễn trong bảng sau:

	Số thiết bị đang được cấp	**Số thiết bị lớn nhất có thể cần**
Người dùng 1	1	4
Người dùng 2	4	6
Người dùng 3	5	8
Dự trữ còn lại		2

Trạng thái đó là ổn định vì cả 3 người dùng có khả năng kết thúc công việc.

2. THUẬT TOÁN BANKER

ƯU ĐIỂM – NHƯỢC ĐIỂM

- **Ưu điểm:**
  - Cho phép cấp phát tài nguyên;
  - Tránh tình trạng Deadlock;
  - Cho phép tiếp tục thực hiện các tiến trình mà trong trường hợp dùng các biện pháp ngăn chặn thì chúng đã bị dừng.
- **Nhược điểm:**
  - Giả thiết số tài nguyên là cố định > <số tài nguyên luôn thay đổi;
  - Số người dùng là không đổi → yêu cầu này không thực tế;
  - Bộ phận phân phối tài nguyên phải đảm bảo thoả mãn tất cả các yêu cầu sau khoảng thời gian hữu hạn → Cần những con số cụ thể hơn.
  - Người dùng phải trả lại các tài nguyên được cấp, sau một khoảng thời gian nào đó → cần các chỉ số cụ thể.
  - Người dùng phải báo trước số lượng lớn nhất tài nguyên anh ta cần → khó đánh giá yêu cầu lớn nhất.

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

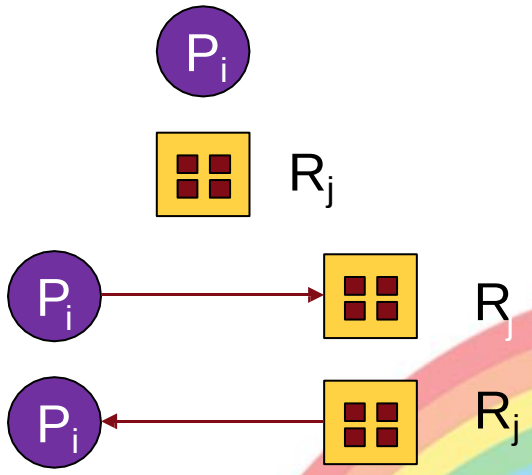
- Phát hiện Deadlock là:
  - Xác định sự kiện xuất hiện trạng thái deadlock;
  - Xác định các tiến trình và tài nguyên nằm trong tình trạng Deadlock;
  - Các thuật toán xác định deadlock thường được áp dụng trong các hệ thống có xuất hiện ba điều kiện đầu tiên làm xuất hiện Deadlock;
  - Sử dụng các thuật toán này phải trả giá, đó là chi phí về thời gian máy.
- Các giải thuật phát hiện Deadlock thường sử dụng RAG và được thiết kế cho mỗi trường hợp sau:
  - Mỗi loại tài nguyên chỉ có một thực thể;
  - Mỗi loại tài nguyên có thể có nhiều thực thể.

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3.1. Biểu đồ phân phối tài nguyên RAG (Resource Allocation Graph)

RAG là đồ thị có hướng, tập đỉnh V và tập cạnh E.

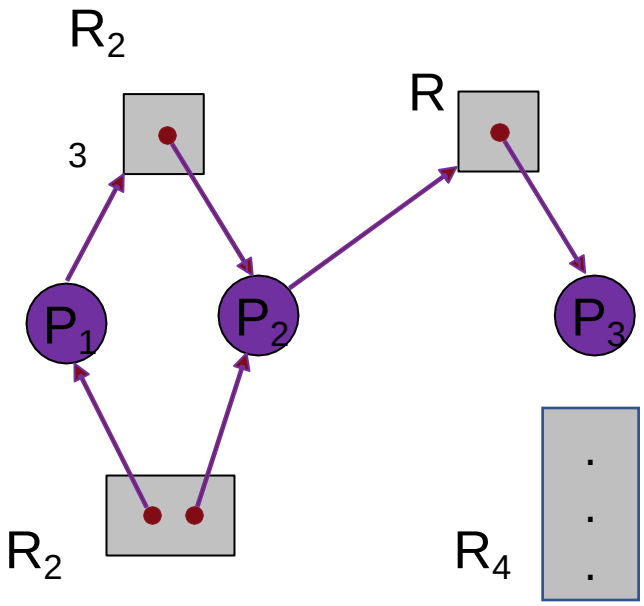
- Tập đỉnh V gồm 2 loại:
  - $P = \{P_1, P_2, \ldots, P_n\}$ (Tất cả tiến trình trong hệ thống).
  - $R = \{R_1, R_2, \ldots, R_m\}$ (Tất cả tài nguyên trong hệ thống)
- Tập cạnh E gồm 2 loại:
  - Request edge: Cạnh có hướng từ $P_i \rightarrow R_j$
  - Assignment edge: Cạnh có hướng từ $R_j \rightarrow P_i$
- Trong đó:
  - P_i: Tiến trình;
  - R_j: Loại tài nguyên với 4 thực thể;
  - P_i: Yêu cầu một thực thể của R_j;
  - P_i: Đang giữ một thực thể cả R_j.



3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3.1. Biểu đồ phân phối tài nguyên RAG (Resource Allocation Graph)

VÍ DỤ VỀ RAG KHÔNG CHU TRÌNH



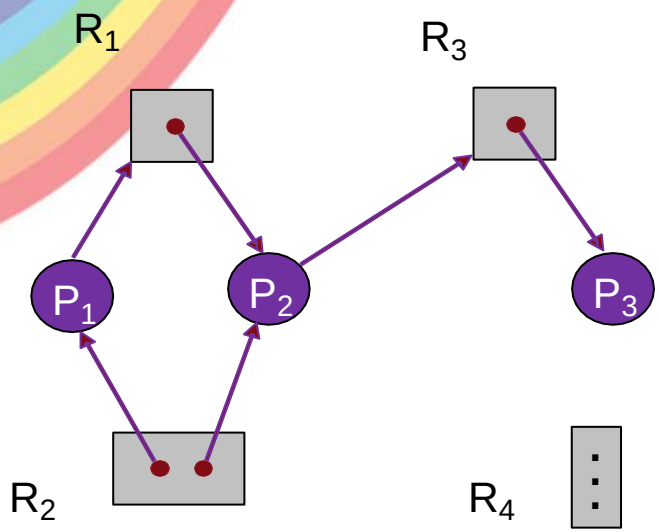
- Nếu đồ thị không chu trình thì sẽ không có tiến trình nào bị Deadlock.
- Nếu đồ thị có chu trình thì có thể tồn tại Deadlock.

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

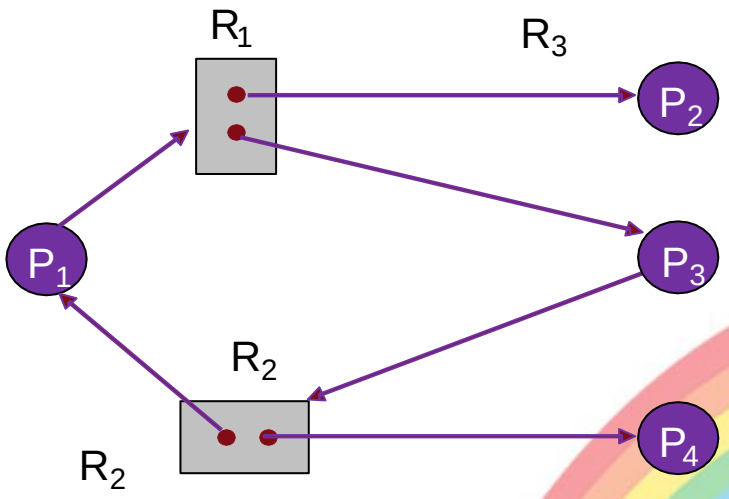
3.1. Biểu đồ phân phối tài nguyên RAG (Resource Allocation Graph)

VÍ DỤ VỀ RAG CÓ CHU TRÌNH

Deadlock



Không Deadlock: P_4 hoặc P_2 có thể kết thúc khiến P_1 và P_3 kết thúc được



Kết luận:

- Nếu RAG không chu trình → Không xẩy ra Deadlock.
- Nếu RAG có chu trình:
  - Nếu mỗi loại tài nguyên chỉ có 1 cá thể chắc chắn xẩy ra Deadlockk
  - Nếu mỗi loại tài nguyên có 1 vài cá thể thì Deadlock có thể xẩy ra.

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3.2. GIẢI THUẬT PHÁT HIỆN DEADLOCK

1. Các biến Work và Finish là vector, kích thước m và n. Khởi tạo:
   - Work:= Available;
   - i = 1, 2,..., n, nếu Allocation n_i # 0 thì Finish[i]:= true.
2. Tìm i thỏa mãn:
   - Finish[i]:= false và thời gian chay của giải thuật O(m – n^2);
   - $Request_i$ <= Work;
   - Nếu không tồn tại i như thế, đến bước 4.
3. Work:= Work + $Allocation_i$
   - Finish[i]:= true;
   - Quay về bước 2.
4. Nếu tồn tại i với Finish[i] = false, thì hệ thống đang ở trạng thái Deadlock. Hơn thế nữa, nếu Finish[i] = false thì Pi bị Deadlock.

Nhận xét: Khi giải thuật phát hiện Deadlock không thấy hệ thống đang Dealock, chưa chắc trong tương lai hệ thống vẫn không Deadlock

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3.2. GIẢI THUẬT PHÁT HIỆN DEADLOCK

VÍ DỤ

Hệ thống có 5 quá trình P0 ,…, P4;

3 loại tài nguyên: A (7 instance), B (2 instance), C (6 instance).

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	0	0
P1	2	0	0	2	0	2			
P2	3	0	3	0	0	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	2			

Hướng dẫn:

- Chạy giải thuật, tìm được chuỗi {P0, P2, P3, P1, P4}
- Với Finish[i] = true, i = 1,…, n, vậy hệ thống không bị Deadlock.

3. PHÁT HIỆN TẮC NGHẼN (DEADLOCK)

3.3. KHÔI PHỤC SAU DEADLOCK

- Trong các hệ thống hiện nay việc khôi phục thường được thực hiện bằng cách loại một số tiến trình để có thể sử dụng các tài nguyên của chúng.
- Có lẽ phương pháp khôi phục tốt nhất là cơ chế tạm dừng/ khởi động tiến trình (Suspend/ Activate).

TÓM LƯỢC CUỐI BÀI

- Trình bày các khái niệm về tiến trình;
- Trình bày về đồng bộ và giải quyết tranh chấp;
- Trình bày về Deadlock, phát hiện Deadlock, ngăn chặn Deadlock, khôi phục sau Deadlock,…