Bài 3: TIẾN TRÌNH VÀ LUỒNG

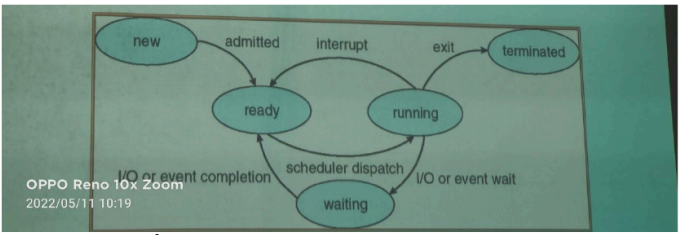
3.1 Tiến trình là gì

- Tiến trình: một thể hiện của việc thi hành một chương trình.
  -Thường gọi là "Heavy Weight Process"
- Tiến trình là một sự trừu tượng hóa cung cấp bởi hệ điều hành, chỉ ra những gì là cần thiết để thi hành một chương trình.
  -Một ngữ cảnh tính toán tách biệt cho mỗi ứng dụng
- Ngữ cảnh tính toán
  -Trạng thái CPU + không gian địa chỉ + môi trường

Thao tác trên tiến trình

- Tạo tiến trình:
  -Khởi động hệ thống
  -Người dùng kích hoạt một chương trình
  -Một tiến trình tạo một tiến trình khác
    Unix/Linux: **exec(), fork()**
    Windows: **CreateProcess()**
  -Cây tiến trình
    Unix/Linux: Các tiến trình cha, con có mỗi quan hệ chặt chẽ
    Windows: các tiến trình cha,
- Dừng tiến trình
  -Xử lý xong lệnh cuối cùng hay gọi lệnh kết thúc
    Unix/Linux: **exit()**
    Windows: **ExitProcess()**
  -Một tiến trình yêu cầu dừng một tiến trình khác
    Unix/Linux: **kill()**
    Windows: **TerminateProcess()**
  -Điều gì xảy ra nếu tiến trình "nạn nhân" vẫn chưa muốn " chết"?
  =>Do lỗi chương trình

Lưu đồ trạng thái của tiến trình



Trạng thái của tiến trình

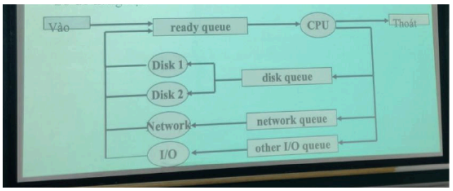
-new: Tiến trình vừa được tạo (chạy chương trình)
-ready: Tiến trình sẵn sàng để chạy (đang chờ cấp CPU)
-running: Tiến trình đang chạy (thi hành lệnh)

Khối điều khiển tiến trình (PCB)

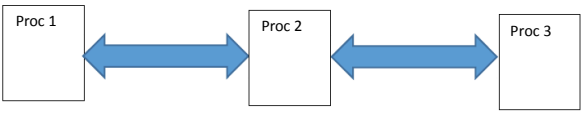
- Ngữ cảnh tiến tính toán của mỗi tiến trình được lưu trong một ***khối điều khiển tiến trình (process control block: PCB)***
- Thông tin gắn với mỗi tiến trình:
  -Trạng thái tiến trình
  -Con trỏ chương trình
  -CPU register
  -Thông tin lập dịch CPU
  -Thông tin quản lý bộ nhớ
  -Thông tin kế toán (ai đang sử dụng bao nhiêu resource)
  -Thông tin trạng thái I/O

Hàng đợi (queue) tiến trình

- Tiến trình khi không thực hiện, được đặt vào hàng đợi
- Các loại:
  -Job queue - tất cả các tiến trình trong hệ thống
  -Ready queue - các tiến trình đang ở trong bộ nhớ và sẵn sàng thực thi
  -Device queue - các tiến trình đang chờ thiết bị I/O
- Các tiến trình di chuyển giữa các queue, không cố định
- Sơ đồ hàng đợi :



Nhiều tiến trình hợp tác



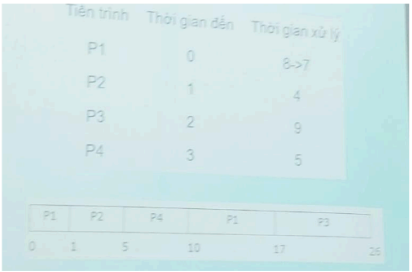
FIRST COMES FIRST SERVICE

Điểm yếu
Giả sử vào hàng đợi theo thứ tự: P2, P3, P1
Sơ đồ Gantt:

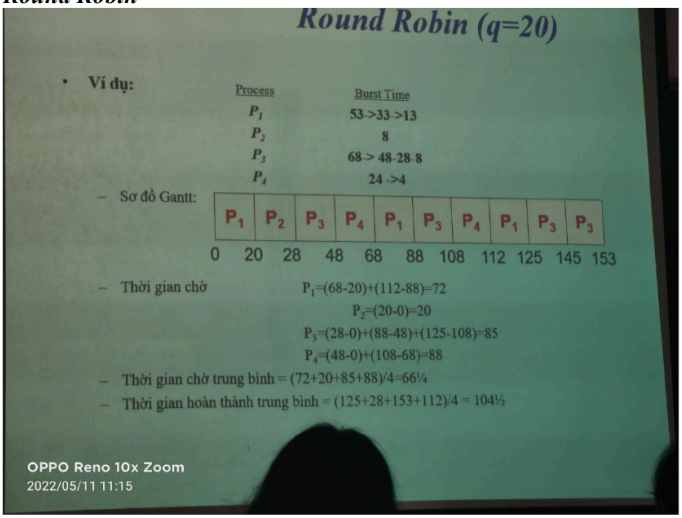
Thời gian chờ P1 = 6, P2 = 0, P3 = 3
Thời gian chờ trung bình: (6 + 0 + 3)/3 = 3
Thời gian hoàn thành trung bình: (3+6+30)/3 = 13
Trường hợp 2:
-Thời gian chờ trung bình tốt hơn (3<17)

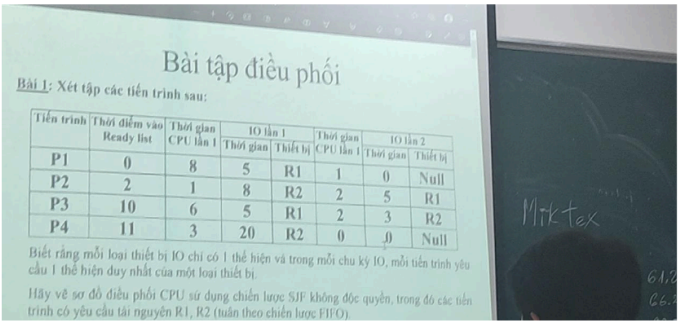
-Thời gian hoàn thành trung bình tốt hơn (13<27)

SHORTEST JOB FIRST



Round Robin





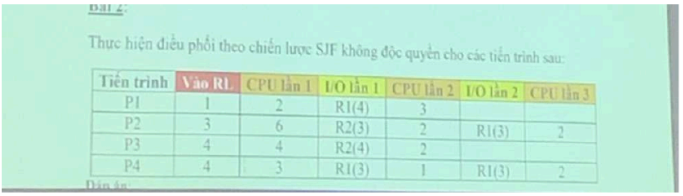
Bài tập điều phối

Bài 1: Xét tập các tiến trình sau:

Tiến trình	Thời điểm vào Ready list	Thời gian CPU lần 1	IO lần 1		Thời gian CPU lần 1	IO lần 2	
			Thời gian	Thiết bị		Thời gian	Thiết bị
P1	0	8	5	R1	1	0	Null
P2	2	1	8	R2	2	5	R1
P3	10	6	5	R1	2	3	R2
P4	11	3	20	R2	0	0	Null

Biết rằng mỗi loại thiết bị IO chỉ có 1 thể hiện và trong mỗi chu kỳ IO, mỗi tiến trình yêu cầu 1 thể hiện duy nhất của một loại thiết bị.

Hãy vẽ sơ đồ điều phối CPU sử dụng chiến lược SJF không độc quyền, trong đó các tiến trình có yêu cầu tài nguyên R1, R2 (tuân theo chiến lược FIFO).



Bài 2:

Thực hiện điều phối theo chiến lược SJF không độc quyền cho các tiến trình sau:

Tiến trình	Vào RL	CPU lần 1	I/O lần 1	CPU lần 2	I/O lần 2	CPU lần 3
P1	1	2	R1(4)	3		
P2	3	6	R2(3)	2	R1(3)	2
P3	4	4	R2(4)	2		
P4	4	3	R1(3)	1	R1(3)	2

Đáp án:

LUỒNG

Khái niệm:

- Luồng (thread) là một dòng điều khiển trong phạm vi một tiến trình.
- Tiến trình đa luồng gồm nhiều dòng điều khiển khác nhau trong cùng không gian địa chỉ.
- Những ưu điểm của đa luồng gồm đáp ứng nhanh đối với người dùng, chia sẻ tài nguyên trong tiến trình, tính kinh tế và khả năng thuận lợi trong khiến trúc đa xử lý.
- Tách biệt:
  - -Trạng thái CPU, ngăn xếp
- Chia sẻ:
  - -Những thứ khác: Data, Code, Heap, môi trường
  - -Đặc biệt: không gian địa chỉ(Tại sao?)
- Mỗi tiến trình luôn có một luồng chính(dòng xử lý cho hàm main())
- Ngoài luồng chính, tiến trình có thể có nhiều luồng con khác.
- Các luồng của một tiến trình
- -Chia sẻ không gian vùng code và data
- -Có vùng stack riêng

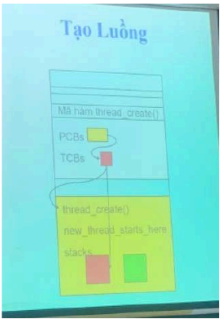
- MultiThreading = một chương trình được tạo ra bằng một số các hoạt động đồng thời
- HeaveWeight Process = Tiến trình với duy nhất một luồng.

KHÁI NIỆM KHỐI QUẢN LÝ LUỒNG (THREAD CONTROL BLOCK - TCB)

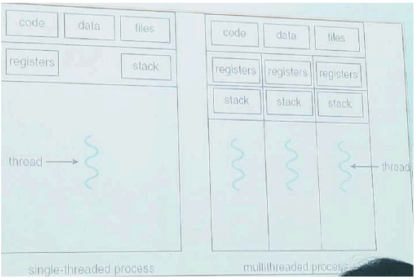
- TCB thường chứa các thông tin riêng của mỗi luồng:
  - ID của luồng
  - Không gian lưu các thanh ghi
  - Con trỏ tới các vị trí xác định trong ngăn xếp
  - Trạng thái của luồng
- Thông tin chia sẻ giữa các luồng trong một tiến trình:
  - Các biến toàn cục
  - Các tài nguyên sử dụng như tập tin, ...
  - Các tiến trình con
  - Thông tin thống kê
  - ...

TẠO LUỒNG

PC
SP



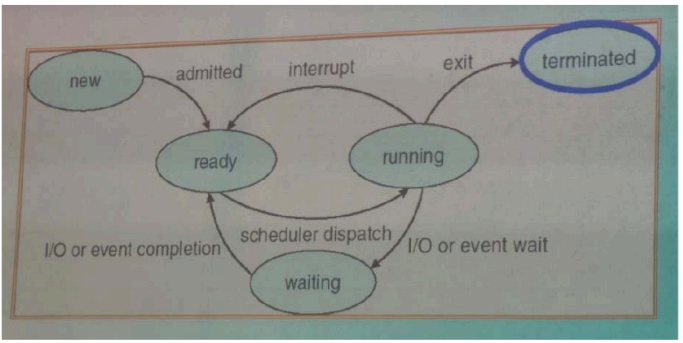
ĐƠN LUỒNG-ĐA LUỒNG



VÍ DỤ VỀ CHƯƠNG TRÌNH ĐA LUỒNG

Database server:
-Nhiều kết nối và cơ sở dữ liệu cùng 1 lúc
Network server:
-Truy cập đồng thời từ môi trường mạng
-Một tiến trình - nhiều thao tác đồng thời
-File server, Web server, ...
Parallel Programming (có nhiều CPU)
-Chia chương trình thành nhiều thread để tận dụng nhiều CPU
-Còn gọi là Multi-Processing

CHUYỂN ĐỔI TRẠNG THÁI CỦA THREAD



- Tương tự như tiến trình:
  -new: Luồng được tạo mới
  -ready: Luồng đang chờ để chạy
  -running: Luồng đang được thi hành

-waiting: Luồng đang chờ sự kiện
-terminated: Luồng kết thúc thi hành

- Thông tin luồng lưu trong TCB

VẤN ĐỀ DEADLOCK

Trong môi trường multiprogramming 1 số process có thể tranh nhau 1 số tài nguyên hạn chế.
1 process yêu cầu các tài nguyên. Nếu tài nguyên không thể đáp ứng tại thời điểm đó thì process sẽ chuyển sang trạng thái chờ.
Các process chờ có thể sẽ không bao giờ thay đổi lại trạng thái được vì cá tai

Ví dụ: trạng thái an toàn

	MAX		CHIẾM		CÒN	
	R1	R2	R1	R2	R1	R2
P1	3	2	1	0		
P2	6	1	2	1	4	1
P3	3	1	2	1		

<P2, P3, P1> hoặc <P2, P1, P3> là trạng thái an toàn

Cột *Max* chỉ số lượng tối đa của mỗi loại tài nguyên mà mỗi tiến trình yêu cầu.
Cột *Chiếm* chỉ số lượng mỗi loại tài nguyên mà mỗi tiến trình đang chiếm giữ (tức là đã được cấp)
Cột *Còn* chỉ số lượng mỗi loại tài nguyên còn rảnh rỗi trong hệ thống, có thể cấp ngay cho các tiến trình có yêu cầu

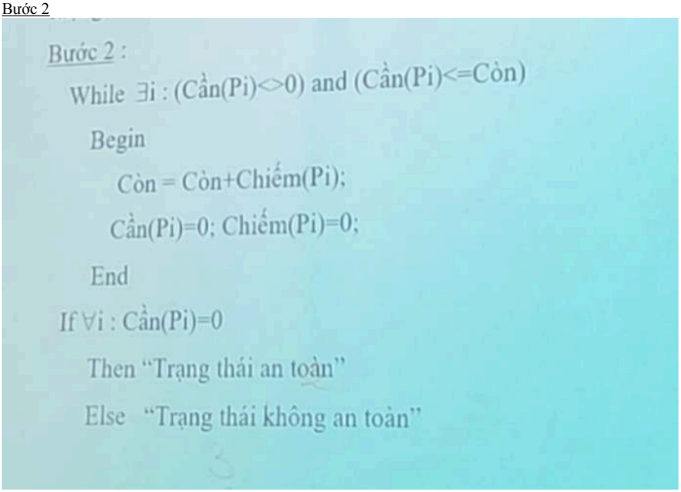
GIẢI THUẬT NHÀ BĂNG

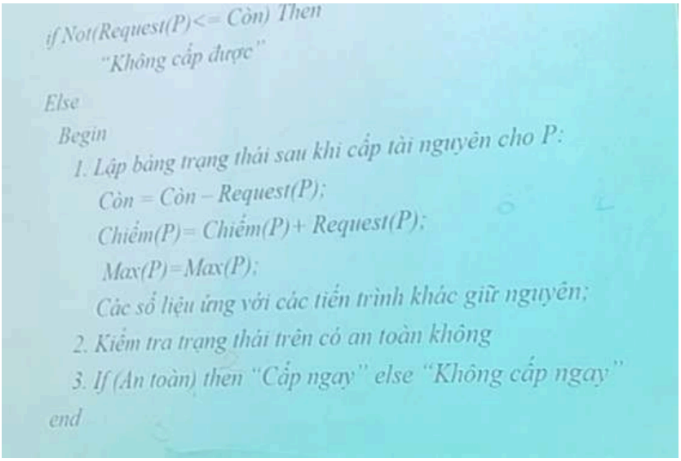
Khi các tiến trình mới được đưa vào hệ thống, nó phải khai báo số tối đa các thể hiện của mỗi loại tài nguyên mà nó cần. Số này có thể không vượt quá tổng số tài nguyên trong hệ thống.

Khi một tiến trình yêu cầu cấp phát tài nguyên, hệ thống phải xác định sau khi cấp phát các tài nguyên này hệ thống có vẫn trong trạng thái an toàn hay không. Nếu trạng thái hệ sẽ vẫn là an toàn, tài nguyên sẽ được cấp, ngược lại, tiến trình phải chờ cho tới khi một vài tiến trình giải phóng toàn bộ tài nguyên.

Giải thuật nhà băng dùng để xác định trạng thái hiện tại có an toàn hay không?

Bước 1;
Từ trạng thái lập bảng nhu cầu trong đó thay cột *Max* bằng cột *Cần* với công thức tính toán *Cần = Max - Chiếm* . Cột *Cần* thể hiện số lượng mỗi loại tài nguyên cần cung cấp thêm cho mỗi tiến trình .
Bước 2





BÀI TẬP TẮC NGHẼN

Thuật ngữ:

Bài tập

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	3	0	1	10	7	4			
P1	3	2	1	8	5	3			
P2	2	1	3	6	3	4	6	2	2
P3	0	3	0	9	6	3			
P4	1	1	2	7	4	5			

P1 yêu cầu tài nguyên (2,0,1)

Kiểm tra
2 0 1 < = 6 2 2
=> thỏa yêu cầu

Update P1 = 3 2 1 + 2 0 1 = 5 2 2
Work = 4 2 1
Need(p0) = 10 7 4 - 3 0 1 = 7 7 3 =>False
Need(p1) = 8 5 3 - 5 2 2 =3 3 1 => False
Need(p2) = 6 3 4 - 2 1 3 = 4 2 1 => True
=> Trả lại tài nguyên Work = 6 3 4

Need(p3) = 9 6 3 - 0 3 0 = 9 3 3 =>False
Need(p4) = 7 4 5 - 1 1 2 = 6 3 3 => True

Need(p0) = 7 7 3 => False
Need(p1) = 3 3 1 => True (Work = 3 0 2)
=>Trả lại tài nguyên Work = 11 5 6

Need(p0) = 7 7 3 => False
Need(p3) = 9 3 3 => True (Work = 2 2 3)
=> Trả lại tài nguyên Work = 11 8 6

Need(p0) = 7 7 3 => True (Work = 4 1 3)
=> Trả lại tài nguyên Work = 14 8 7
Need(p4) = 6 3 3 =>True (Work = 8 5 4)
=> Trả lại tài nguyên Work = 15 9 9

=> <P2, P1, P3, P0, P4> là trạng thái an toàn