Chương 2

QUẢN LÝ TIẾN TRÌNH (Process Management)

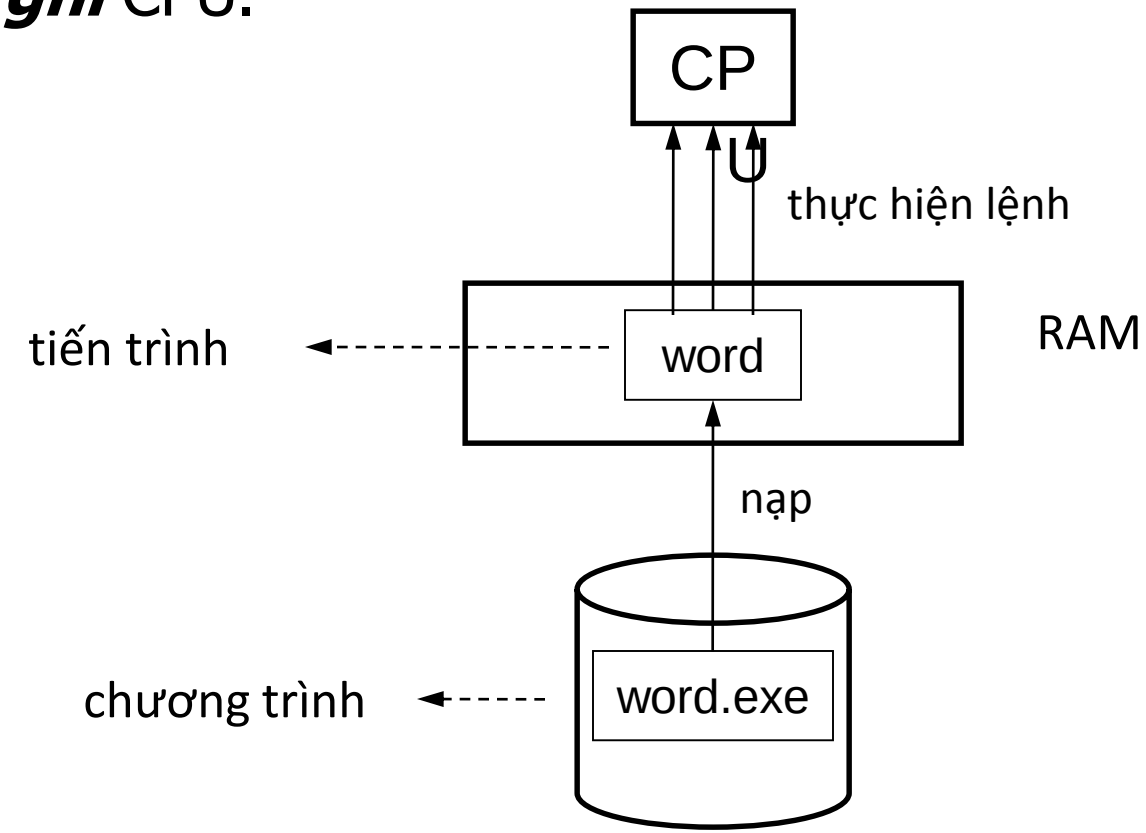
Nội Dung Chương 2

I. Khái niệm tiến trình
II. Đa nhiệm, đa tiến trình
III. Tổ chức điều phối đa tiến trình
IV. Chiến lược điều phối tiến trình
V. Tiểu trình và Mô hình đa tiểu trình

Khái niệm tiến trình

1. *Định nghĩa*:

Tiến trình là một chương trình ***đang được thực thi*** trong máy tính, sở hữu một ***vùng bộ nhớ*** và một tập các ***thanh ghi*** CPU.



Khái niệm (tt)

- CPU burst (chu kỳ CPU): dãy các lệnh chỉ sử dụng CPU
- I/O burst: Khoảng thời gian truy xuất thiết bị nhập xuất.

- Mô hình tiến trình theo chu kỳ CPU và chu kỳ I/O.

CPU	IO	CPU	IO	CPU

- Tiến trình hướng CPU và tiến trình hướng I/O.

Đa nhiệm, đa tiến trình

1) Bản chất CPU là đơn nhiệm:

   tại 1 thời điểm chỉ thực hiện 1 lệnh assembly.

   - MS-DOS (đơn nhiệm): không thể vừa lập trình C++, vừa gõ văn bản Sidekick.

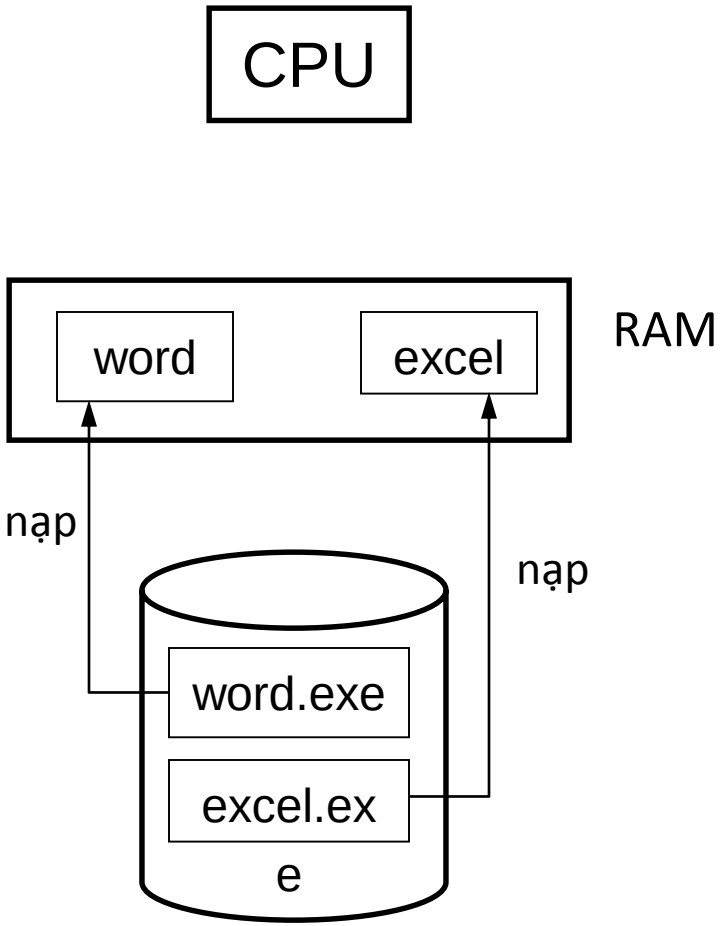
2) Nhu cầu đa nhiệm:

   Đa nhiệm (multitask – multiprogram): nhiều thuận lợi hơn cho người sử dụng. Ví dụ:

   - Windows (đa nhiệm): nhiều chương trình hoạt động cùng lúc Word, Excel, duyệt Web, nghe nhạc.

Cơ chế thực hiện đa nhiệm, đa tiến trình

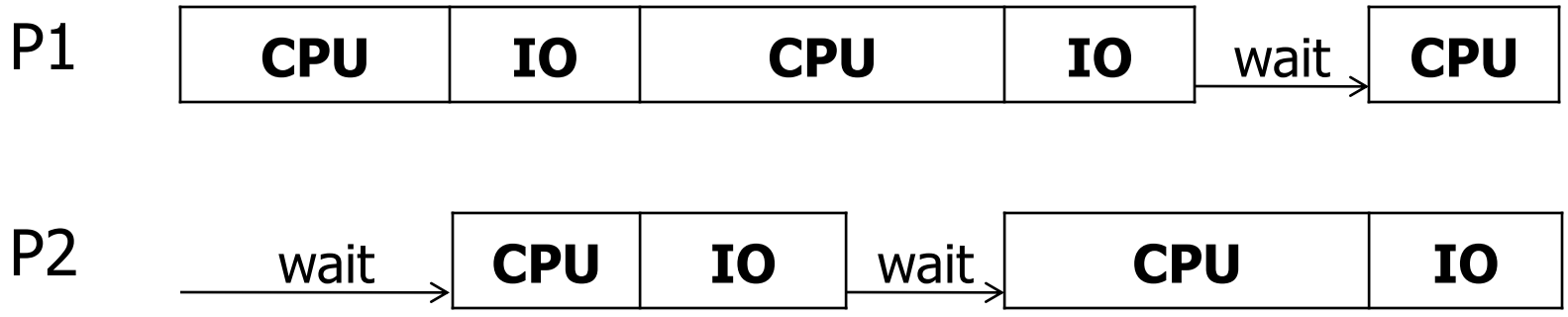
Về mặt bộ nhớ: bộ nhớ có thể nạp nhiều chương trình cùng lúc



Về mặt CPU: Cần sự can thiệp điều phối của hệ điều hành, phân chia đan xen CPU cho các tiến trình.

a) Điều phối non-preemtive (độc quyền):

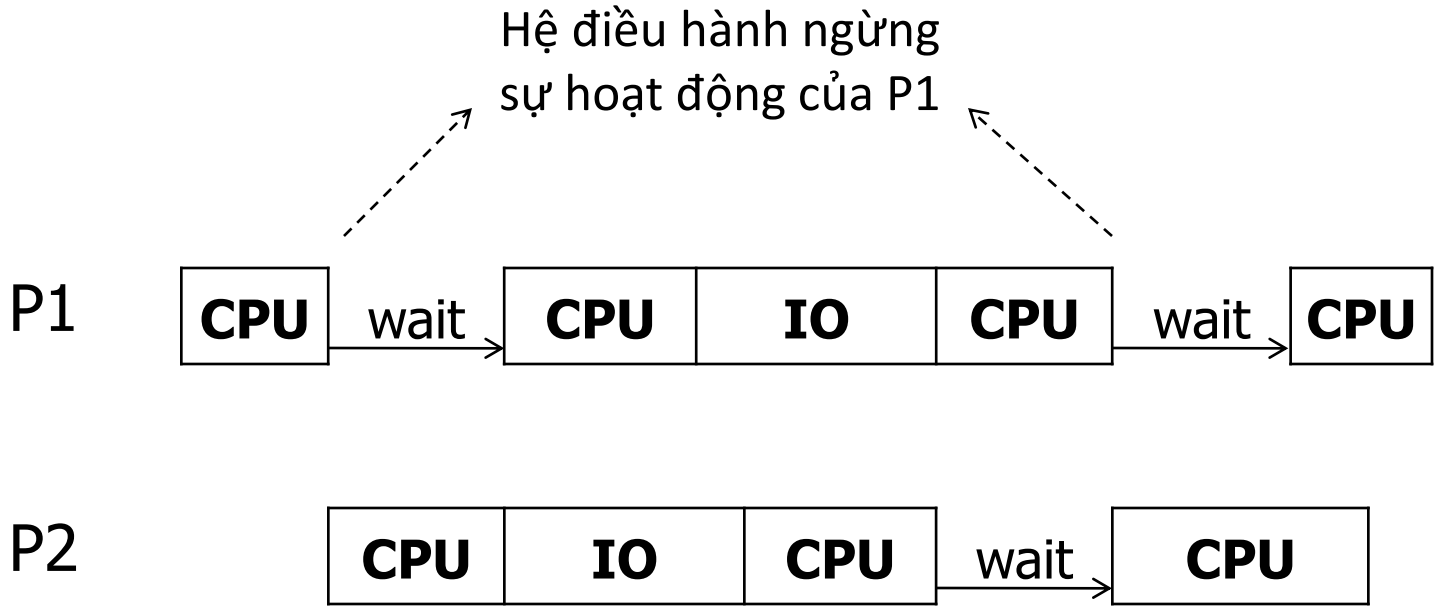
Khi 1 tiến trình kết thúc chu kỳ CPU để chuyển sang chu kỳ IO → nhường CPU cho tiến trình khác.



Ví dụ: Windows 3.1

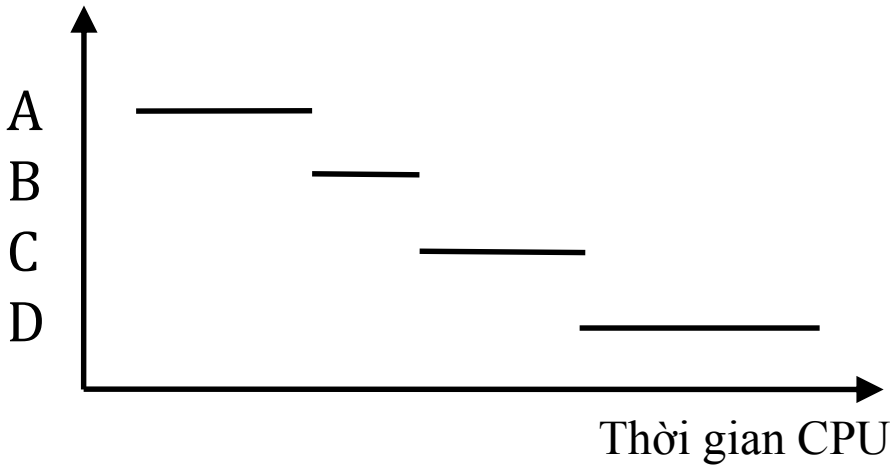
b) <u>Điều phối preemtive (không độc quyền):</u>

Tiến trình có thể bị HĐH thu hồi CPU bất cứ thời điểm nào để nhường cho tiến trình khác.

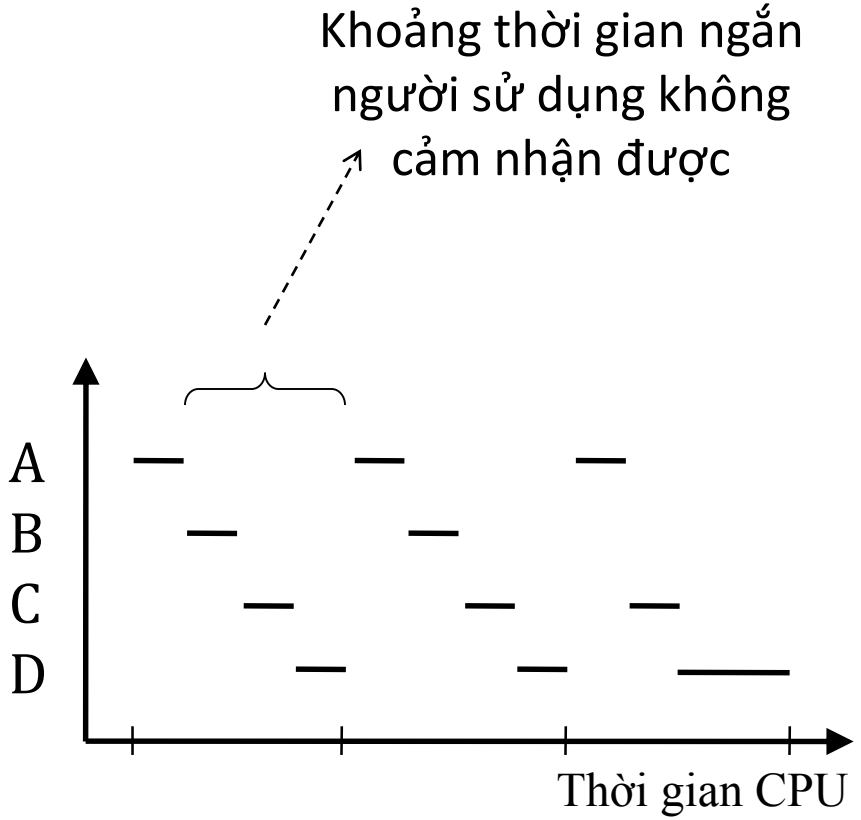


Ví dụ: Windows 95 trở về sau.

So sánh giữa đơn nhiệm và đa nhiệm



a) Mô hình tuần tự các chu kỳ CPU

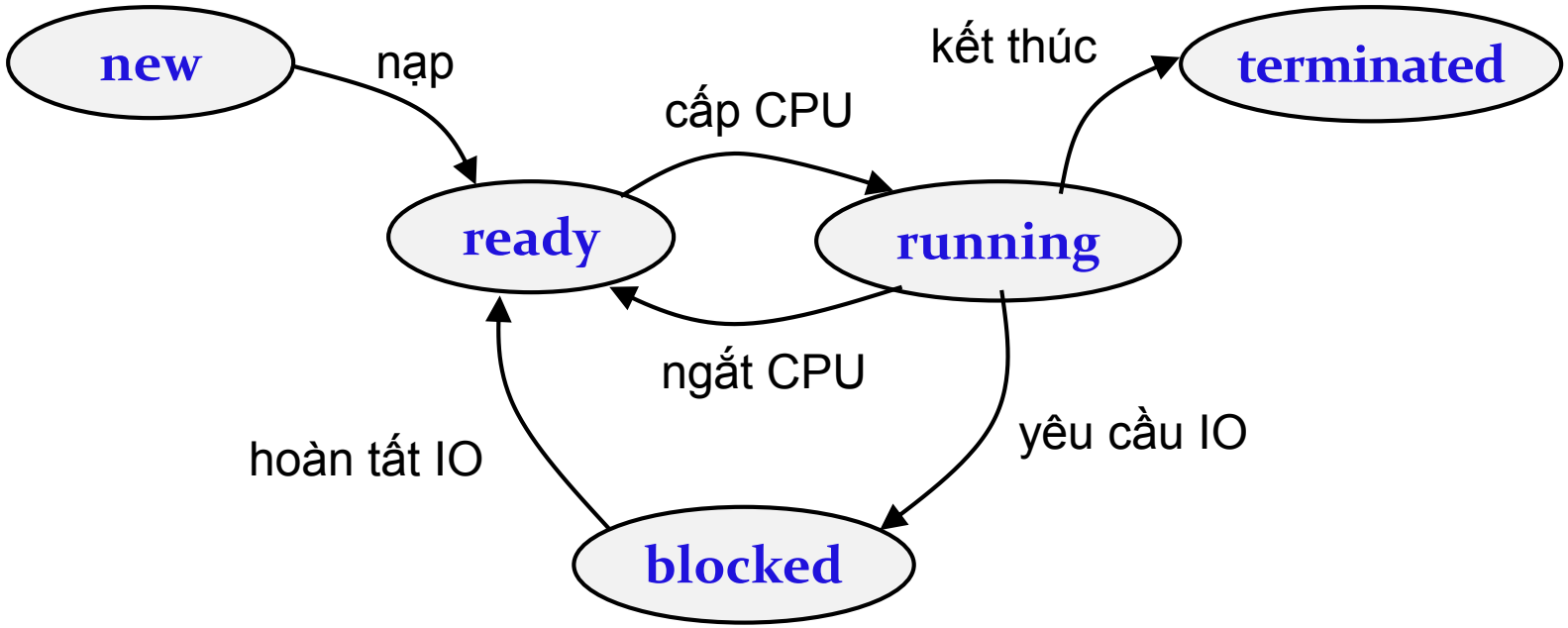


b) Mô hình song song các chu kỳ CPU

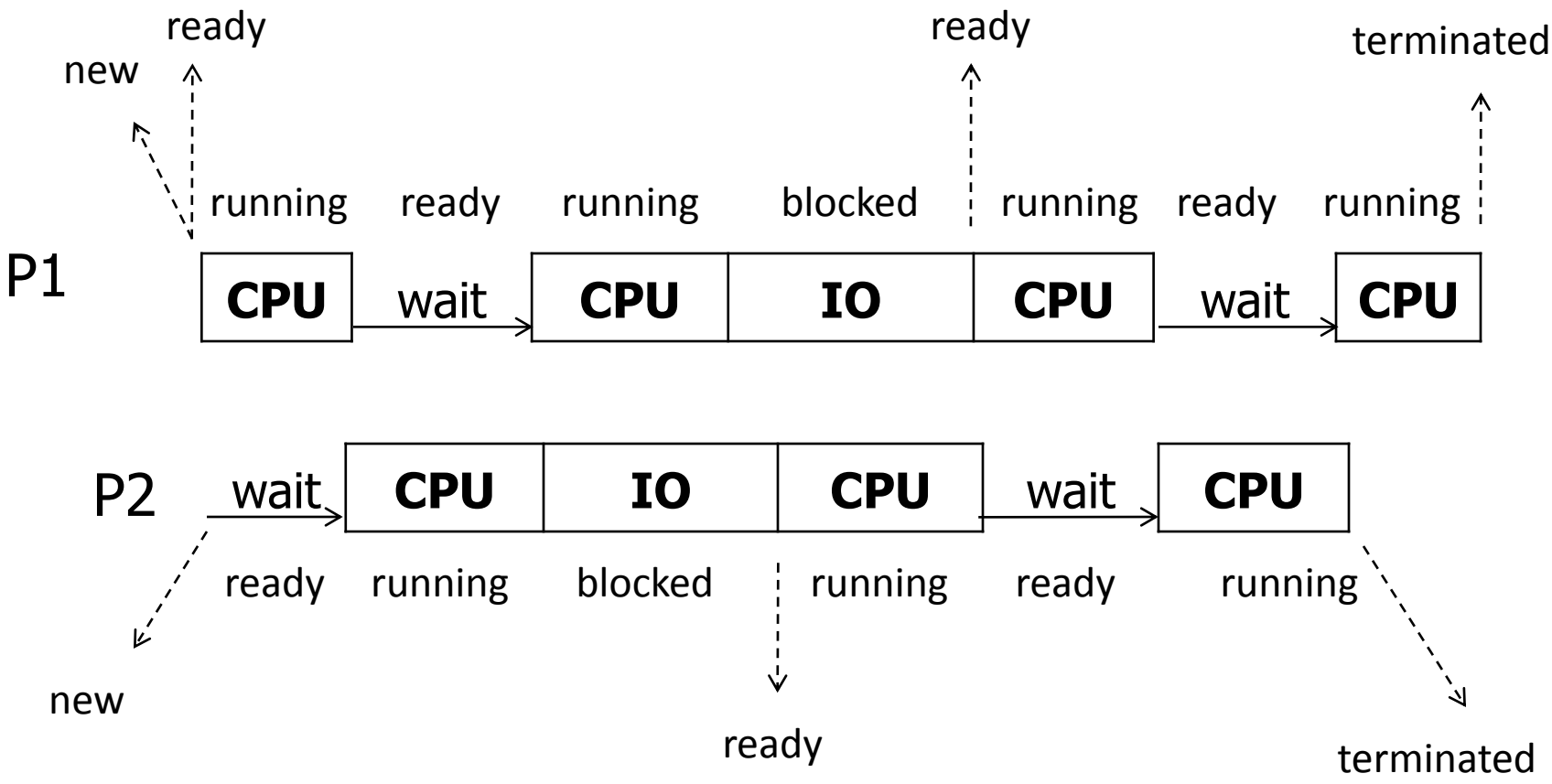
Tổ chức điều phối đa tiến trình

a) Các trạng thái của tiến trình:

- 5 trạng thái.
- Sơ đồ chuyển trạng thái:

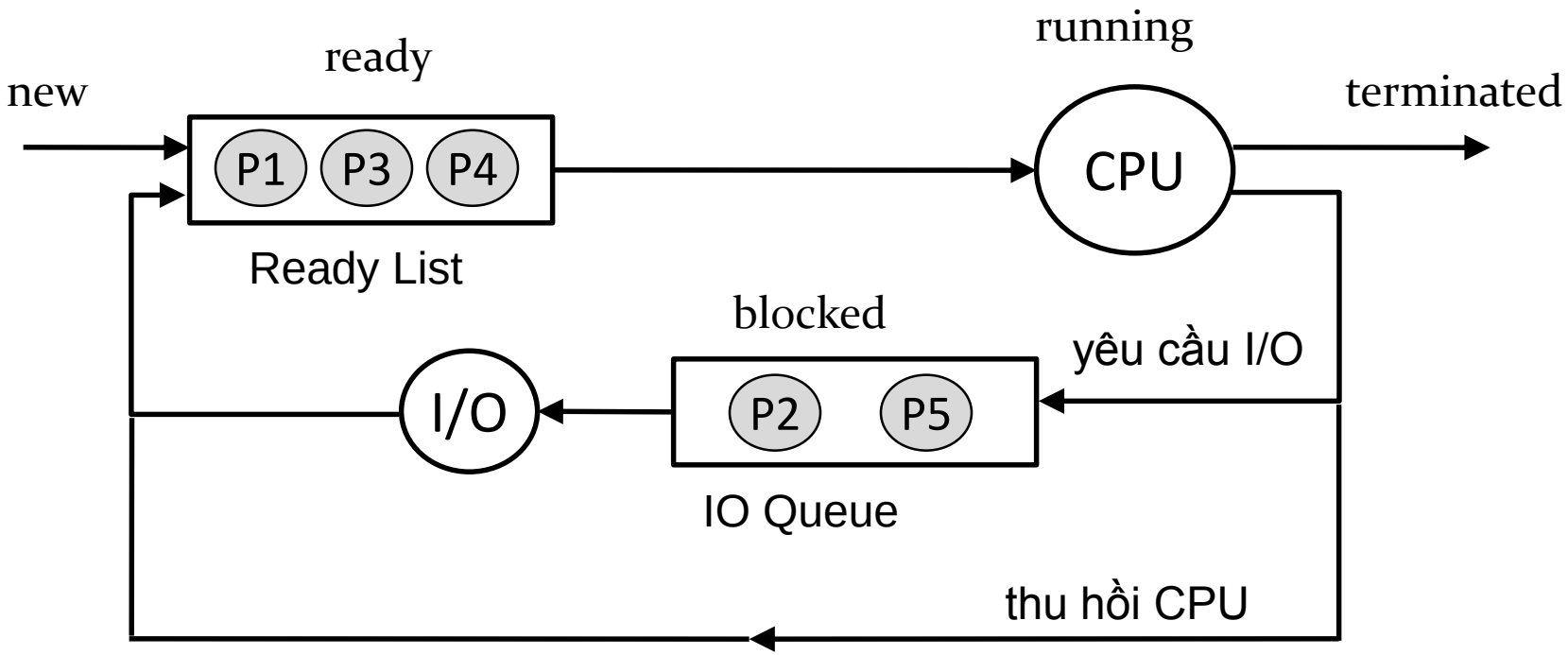


Ví dụ: Điều phối (preemtive không độc quyền):



b) Cơ chế hàng đợi điều phối (Ready List)

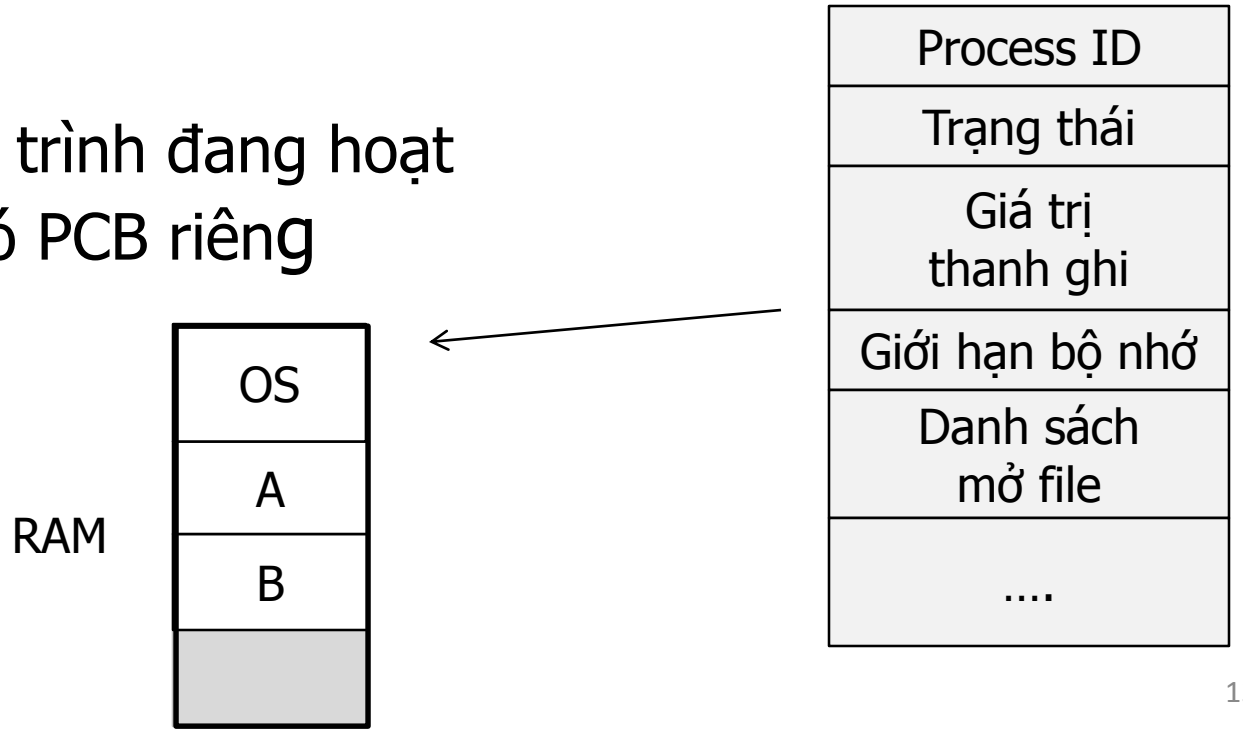
- Tại 1 thời điểm chỉ có 1 tiến trình running
- Các tiến trình ở trạng thái ready chờ trong ReadyList



c) Khối quản lý tiến trình (Process Control Block - PCB)

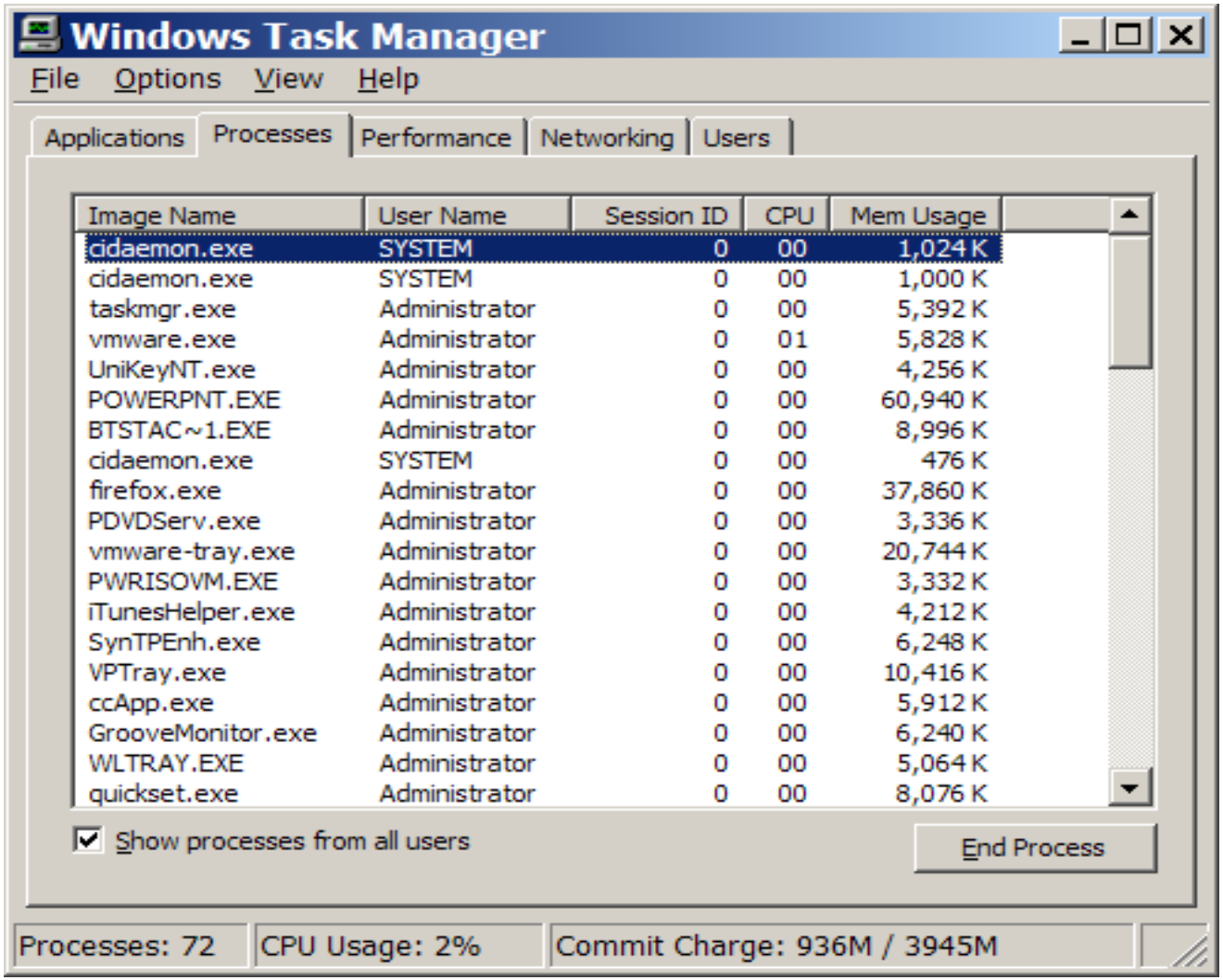
- Hệ điều hành lưu các thông tin về tiến trình trong PCB.
  - Định danh tiến trình (Process ID)
  - Trạng thái tiến trình
  - Giá trị các thanh ghi CPU của tiến trình
  - Danh sách các file đang mở
  - ...

- Mỗi tiến trình đang hoạt động đều có PCB riêng



Quản lý tiến trình: Task Manager and Process Explorer

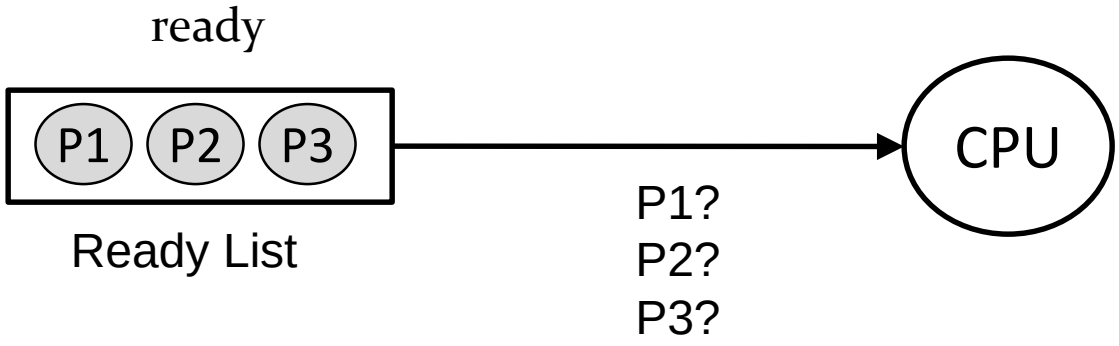
Task Manager



IV. Chiến lược điều phối

1. Định nghĩa chiến lược điều phối

- Là việc quyết định tiến trình nào được cấp CPU, tiến trình nào phải chờ → bảo đảm cho hệ thống hoạt động tối ưu nhất.



2. Mục tiêu điều phối

a. **Sự công bằng (Fairness):**
   Các tiến trình chia sẻ CPU một cách công bằng, không tiến trình nào phải chờ lâu.

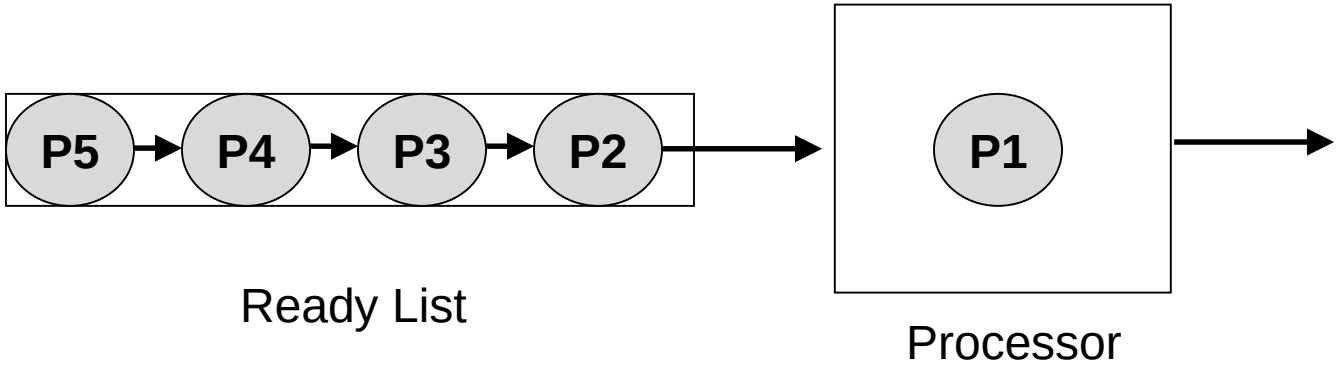
b. Thời gian đáp ứng hợp lý (Response time):
   Cực tiểu hoá thời gian hồi đáp cho các tương tác của người sử dụng

c. Thời gian lưu lại trong hệ thống (Turnaround Time):
   Cực tiểu hóa thời gian hoàn tất các tác vụ xử lý

d. Thông lượng tối đa (Throughput):
   Cực đại hóa số công việc được xử lý trong một đơn vị thời gian

3. Chiến lược công việc đến trước (First Come First Served - FCFS)

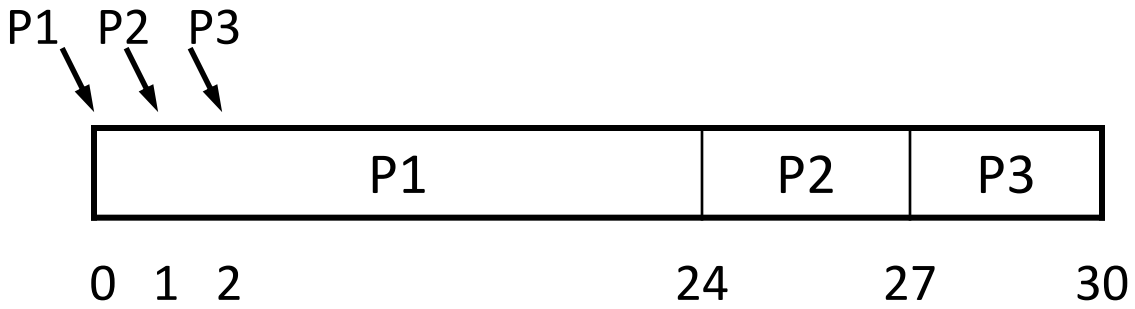
- Tiến trình nào đến trước thì được sử dụng CPU trước
- Thuộc dạng điều phối độc quyền: tiến trình chỉ rời CPU cho đến khi hoàn tất CPU burst

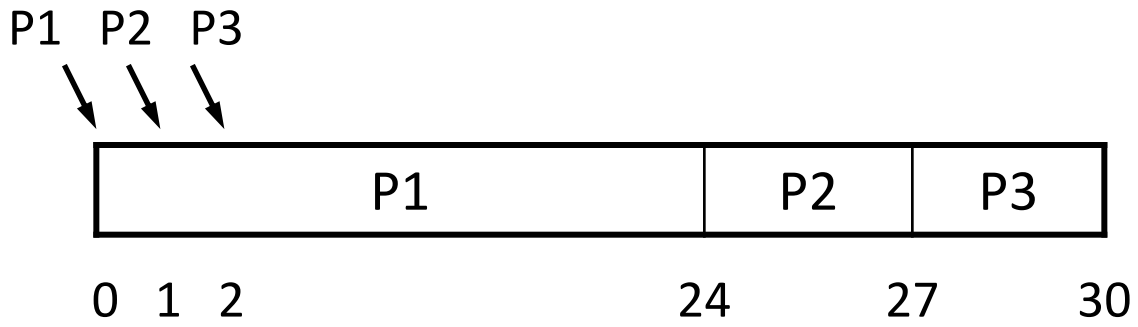


Ví dụ : Cho 3 tiến trình P1, P2, P3 như sau:

Tiến trình	Thời điểm vào ready list	Thời gian xử lý (miligiây)
P1	0	24
P2	1	3
P3	2	3

Thứ tự cấp phát CPU cho 3 tiến trình trên là:





- Thời gian chờ (waiting time)

  P1: 0 ms P2: 24 – 1 = 23 ms P3: 27 – 2 = 25 ms

  → Thời gian chờ trung bình: (0+23+25)/3 = 16 ms

- Nếu thứ tự đến các quá trình là P2 - 0, P3 - 1, P1 - 2 thì thời gian chờ là bao nhiêu?

- Nhận xét :
  - Thời gian chờ phụ thuộc vào thứ tự đến của các tiến trình → không cực tiểu
  - Có thể xảy ra trường hợp nhiều tiến trình phải chờ 1 tiến trình khác với CPU burst dài.

4. Chiến lược công việc ngắn nhất (Shortest Job First – SJF)

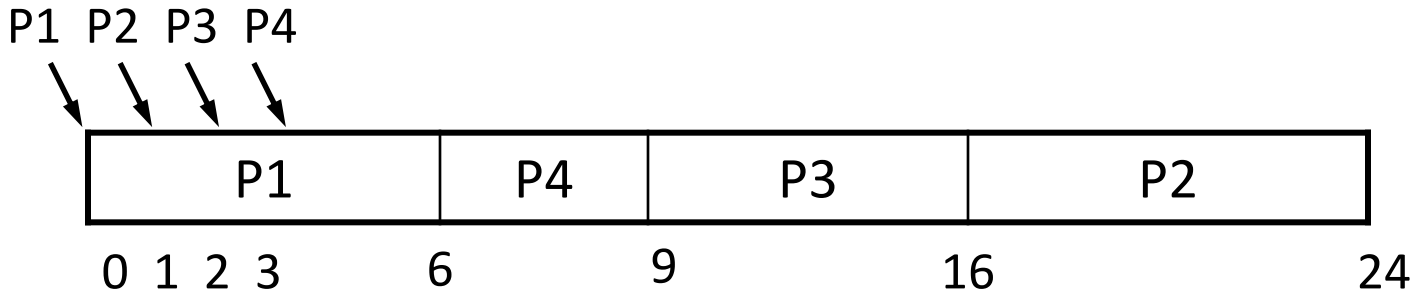
- Tiến trình nào có chiều dài CPU burst tiếp theo ngắn nhất sẽ được cấp CPU.
- Thuộc dạng điều phối độc quyền.

Ví dụ : Cho 4 tiến trình P1, P2, P3, P4 với chiều dài CPU burst **:**

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P1	0	6
P2	1	8
P3	2	7
P4	3	3

P1	0	6
P2	1	8
P3	2	7
P4	3	3

Thứ tự cấp phát CPU:



Thời gian chờ:

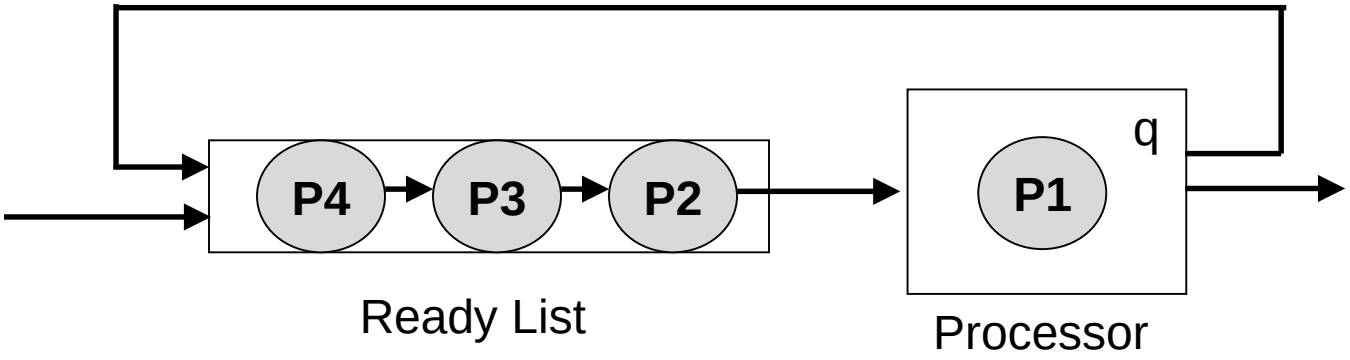
P1: 0 P2: 16-1=15 P3: 9-2= 7 P4: 6 - 3 = 3

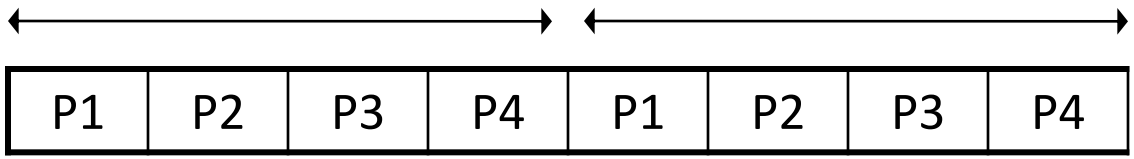
→ Thời gian chờ trung bình: (0+15+7+3)/4=6.25 ms

- Nếu dùng FCFS, thời gian chờ (theo thứ tự P1,P2,P3,P4) là bao nhiêu ?
- Nhận xét: SJF có thời gian chờ là tối ưu.
- Khó khăn: làm sao biết *trước* được chiều dài CPU burst của mỗi tiến trình?

5. Chiến lược phân phối xoay vòng (Round Robin – RR)

- Điều phối dạng không độc quyền, tiến trình có thể bị thu hồi CPU dù chưa thực hiện xong CPU burst
- Mỗi tiến trình được cung cấp một khoảng thời gian (gọi là time quantum - q) để thực hiện. Hết time quantum, bộ điều phối sẽ dừng tiến trình, đưa vào lại Ready List và cấp CPU cho tiến trình tiếp theo.



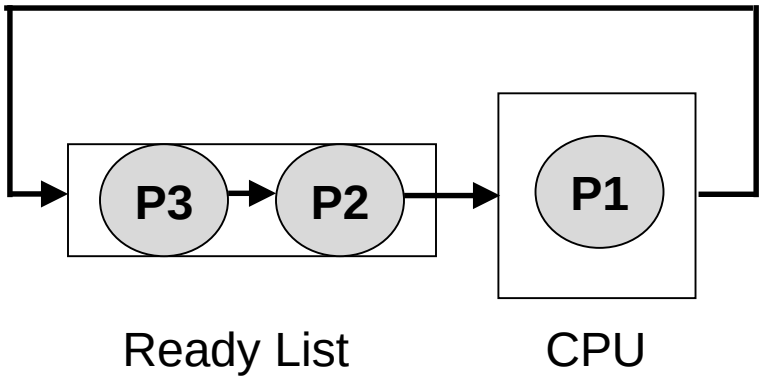


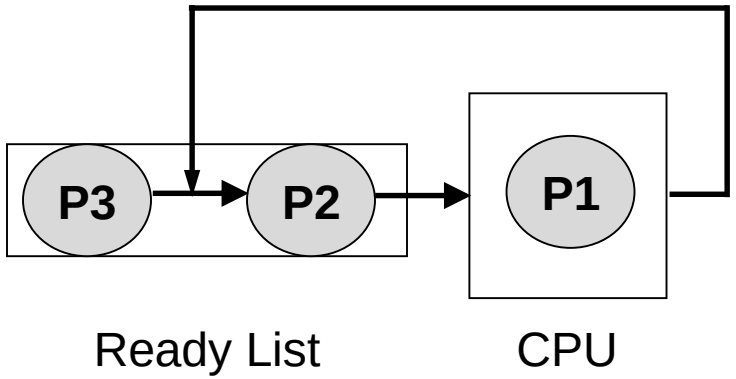
Ví dụ : Cho 3 tiến trình P1, P2, P3 với chiều dài CPU burst :

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P1	0	14
P2	2	4
P3	4	5

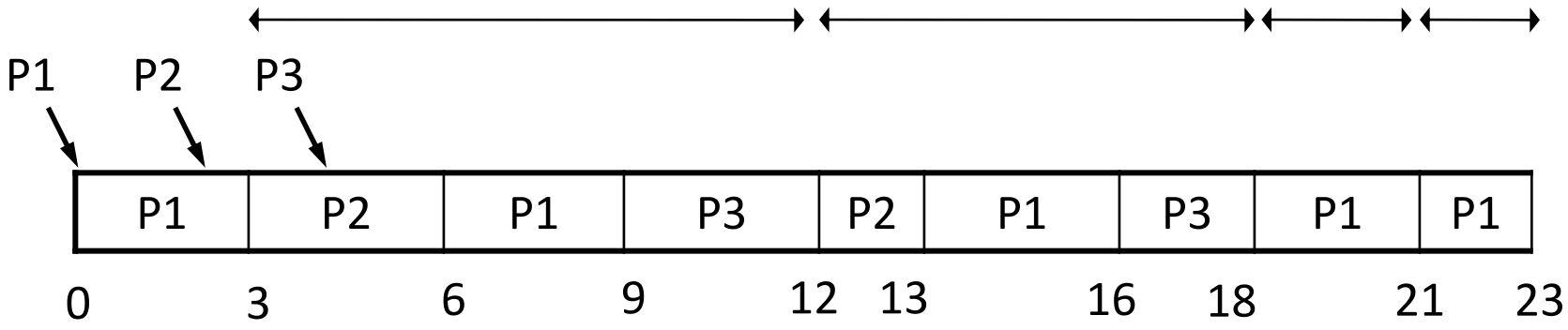
Time quantum q là 3

Câu hỏi: tại thời điểm 3?





Thứ tự ReadyList tại thời điểm 4: P2 ← P1 ← P3



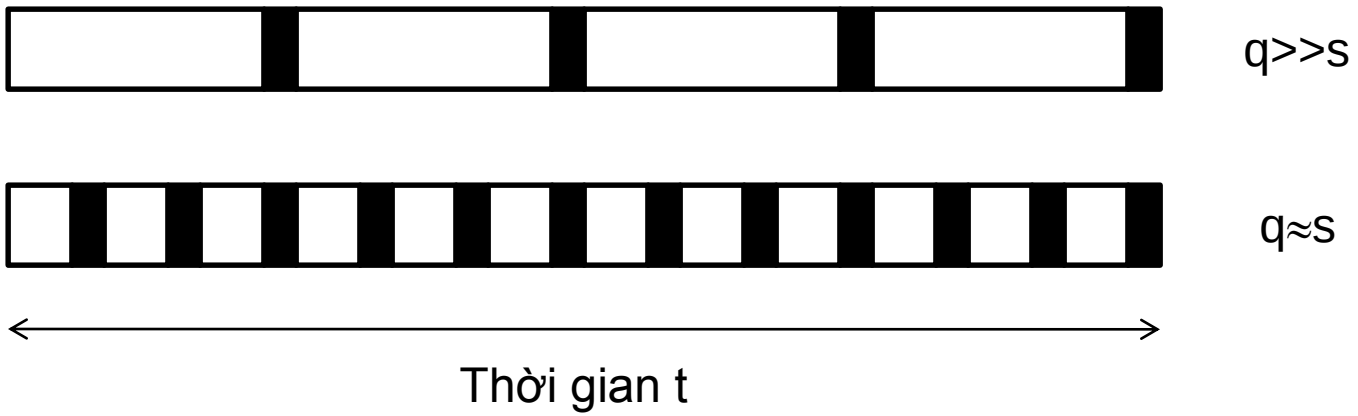
Thời gian chờ:

P1: 0+3+4+2=9 P2: 3-2+6= 7 P3: 9 - 4+4 = 9

→ Thời gian chờ trung bình: 25/3 = 8.33 ms

Nếu theo FCFS thì thời gian chờ là bao lâu?

- Với n tiến trình trong Ready List, quantum q: mỗi tiến trình cần chờ tối đa là (n-1)q để được vào CPU → Không có tình trạng chờ quá lâu.
- Quá trình context switch tốn thời gian là s. Do đó cần có q>>s để việc sử dụng CPU được hiệu quả. Nếu q ≈ s thì thời gian CPU chủ yếu dành cho chuyển đổi tiến trình.



- Tuy nhiên cũng không được chọn q quá dài vì nếu thế RR sẽ trở thành FCFS

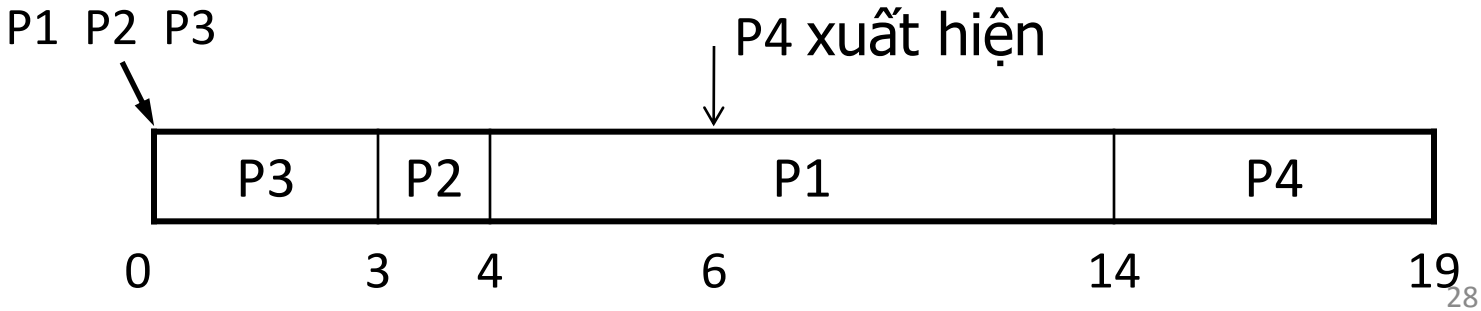
6. Chiến lược điều phối theo độ ưu tiên (Priority- based)

- Mỗi tiến trình được gán một con số gọi là độ ưu tiên.
- Tiến trình nào có số ưu tiên lớn hơn thì sẽ được cấp CPU thực hiện.

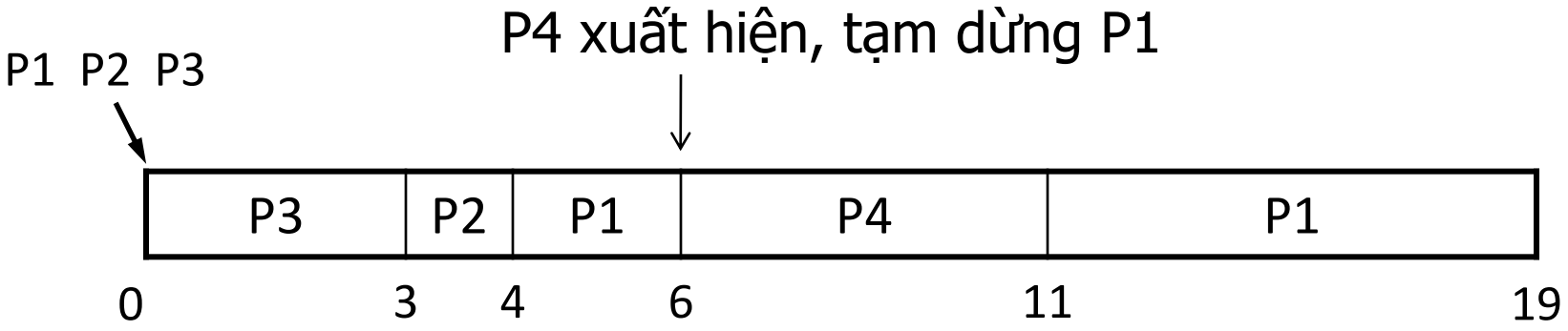
Xét ví dụ:

T.trình	Chiều dài CPU burst	Độ ưu tiên	Thời điểmvào RL
P1	10	4	0
P2	1	2	0
P3	3	1	0
P4	5	3	6

- Nếu điều phối độc quyền:



- Nếu điều phối không độc quyền:

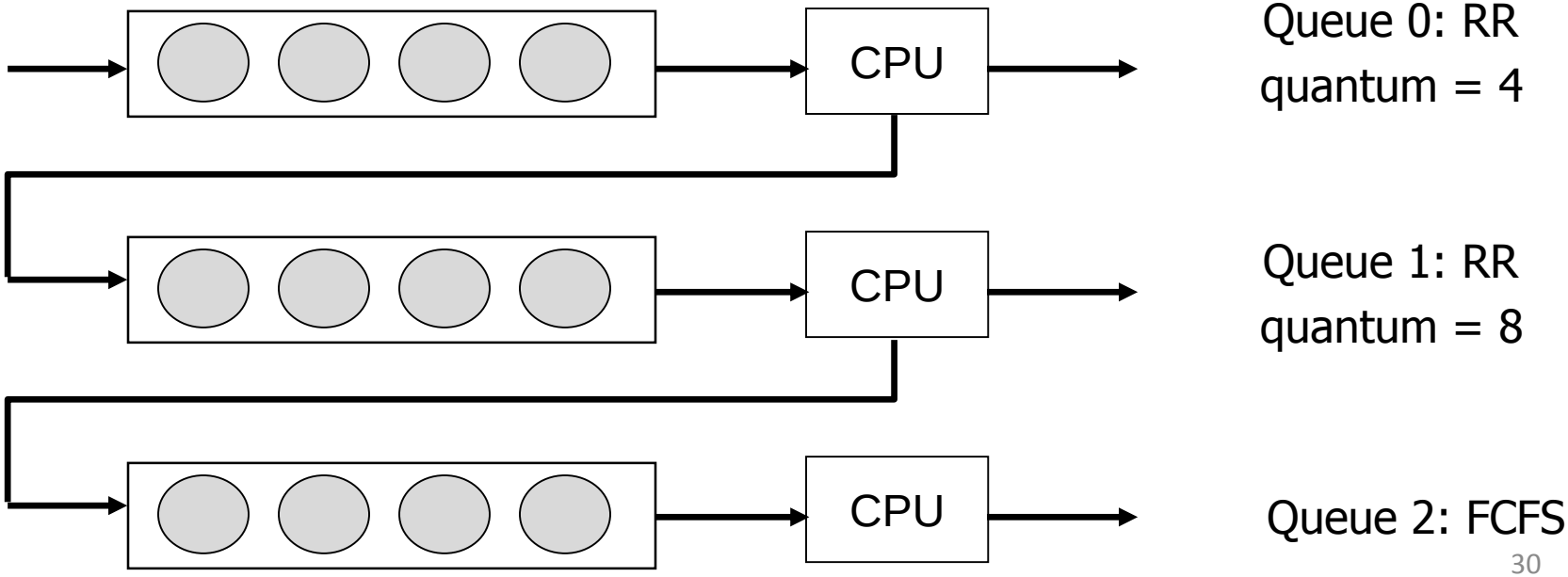


- Nhược điểm của điều phối theo độ ưu tiên: tiến trình với độ ưu tiên thấp có thể chờ mãi mãi

→ Biện pháp: tăng độ ưu tiên sau một thời gian chờ đợi (aging)

7. Chiến lược hàng đa mức hồi tiếp (Multilevel feedback-queue)

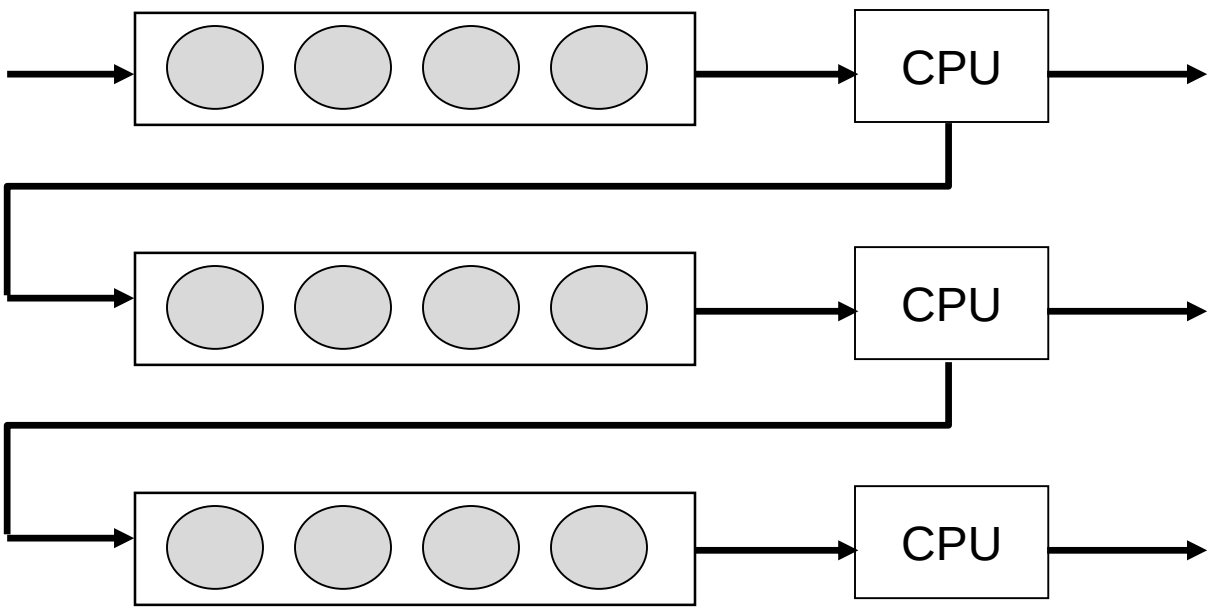
- Ready List được phân thành n danh sách nhỏ hơn (queue). Các queue được đánh số thứ tự ưu tiên từ 0 đến n-1.
- Mỗi queue sẽ áp dụng một chiến lược điều phối khác nhau trong các chiến lược FCFS, SJF, RR. Ví dụ:

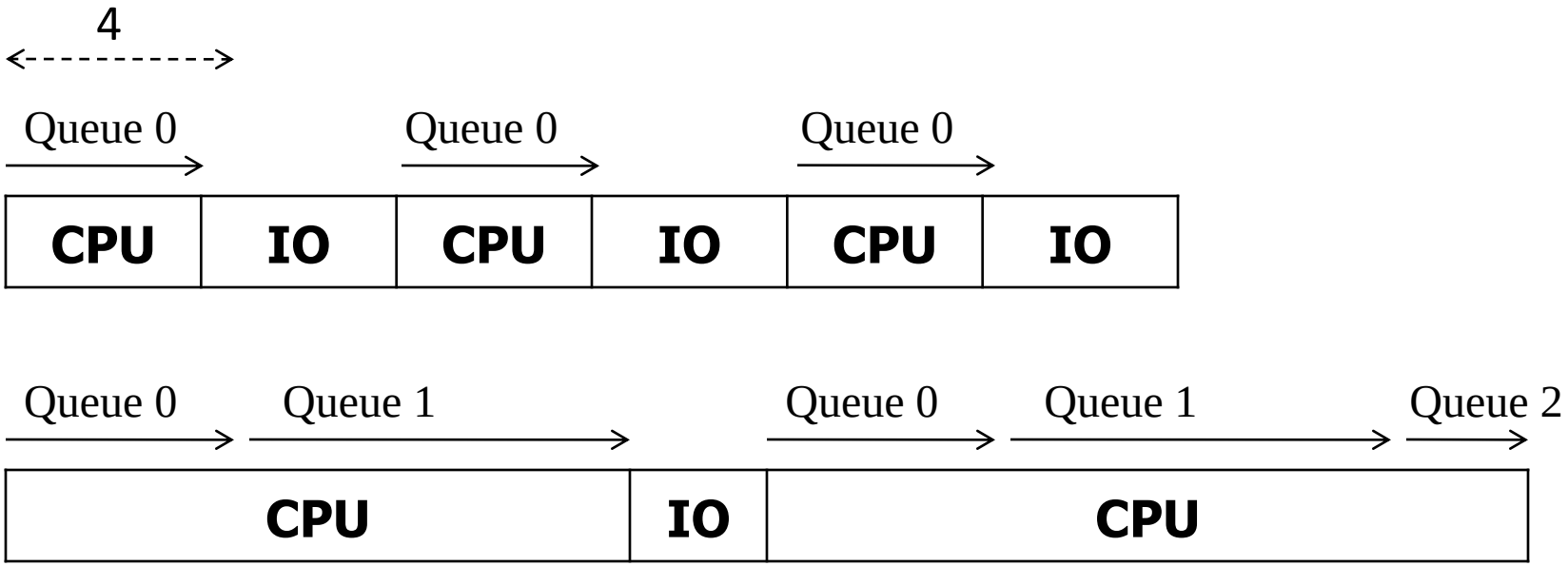


- Một tiến trình khi bắt đầu CPU burst được đưa vào chờ tại queue 0. Sau khi được cấp CPU, nếu chưa thực hiện xong CPU burst thì tiến trình sẽ chuyển xuống chờ tại queue 1. Tương tự như vậy cho các queue khác.
- Queue k+1 chỉ được điều phối khi queue k không còn tiến trình nào chờ.

Mục đích:

- Tiến trình nào dùng nhiều CPU sẽ có độ ưu tiên thấp → dành CPU cho các tiến trình hướng I/O.
- Tiến trình dùng nhiều CPU sẽ điều phối theo RR quantum lớn hay FCFS → giảm context switch.


CPU
CPU
CPU
Queue 0: RR
quantum = 4
Queue 1: RR
quantum = 8
Queue 2: FCFS


4
Queue 0
Queue 0
Queue 0
CPU
IO
CPU
IO
CPU
IO
Queue 0
Queue 1
Queue 0
Queue 1
Queue 2
CPU
IO
CPU

Mục đích:

- Tiến trình nào dùng nhiều CPU sẽ có độ ưu tiên thấp → dành CPU cho các tiến trình hướng I/O.
- Tiến trình dùng nhiều CPU sẽ điều phối theo RR quantum lớn hay FCFS → giảm context switch.

8. Điều phối tiến trình trong Windows

- Dùng chiến lược điều phối theo độ ưu tiên không độc quyền và Round-Robin.
- Các mức ưu tiên:
  - REAL_TIME_PRIORITY_CLASS (24)
  - HIGH_PRIORITY_CLASS (13)
  - ABOVE_NORMAL_PRIORITY_CLASS (10)
  - NORMAL_PRIORITY_CLASS (8)
  - BELOW_NORMAL_PRIORITY_CLASS (6)
  - IDLE_PRIORITY_CLASS (4)

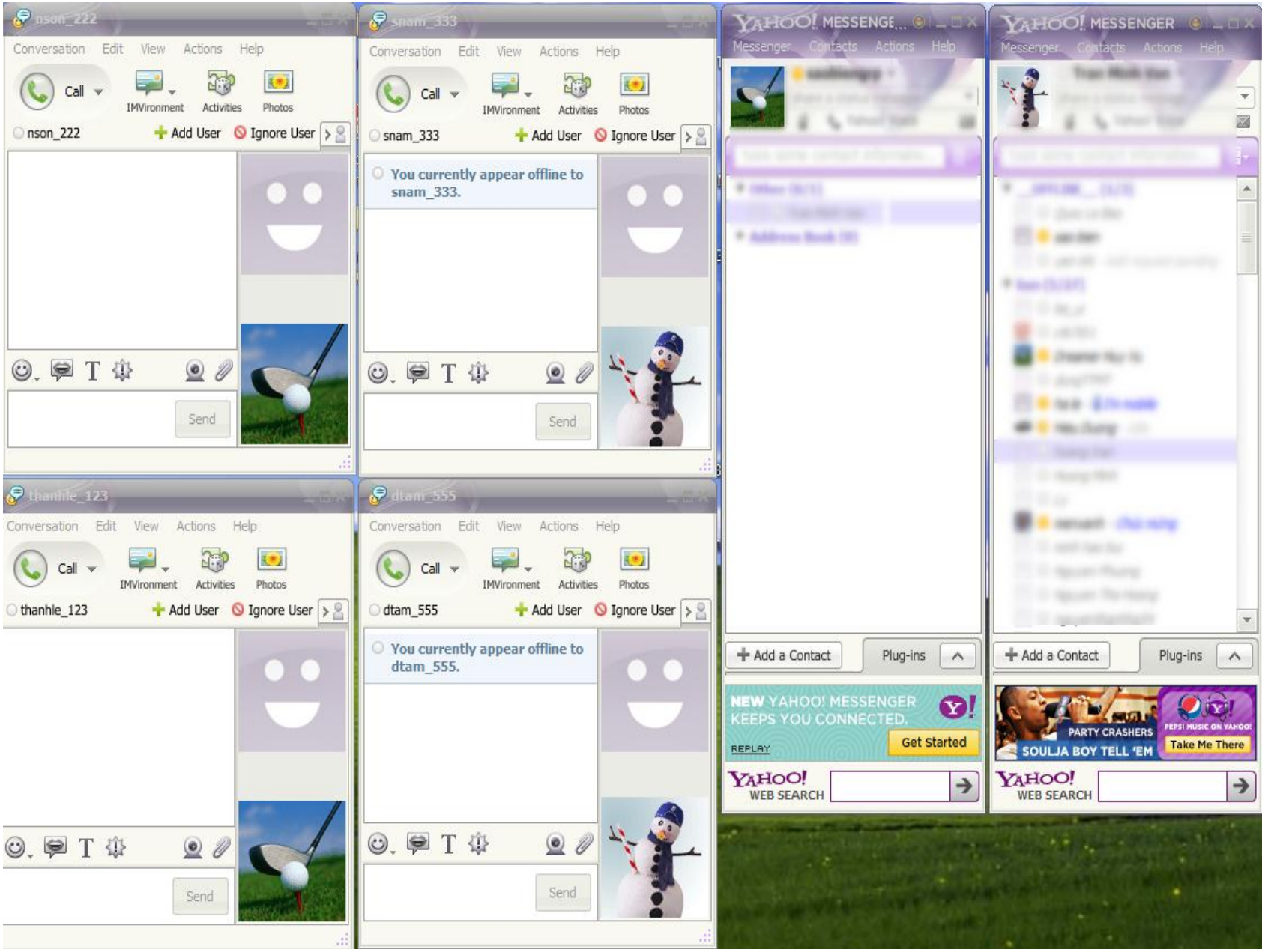
Xem và thay đổ độ ưu tiên bằng Process Explorer.

V. Tiểu trình và đa tiểu trình

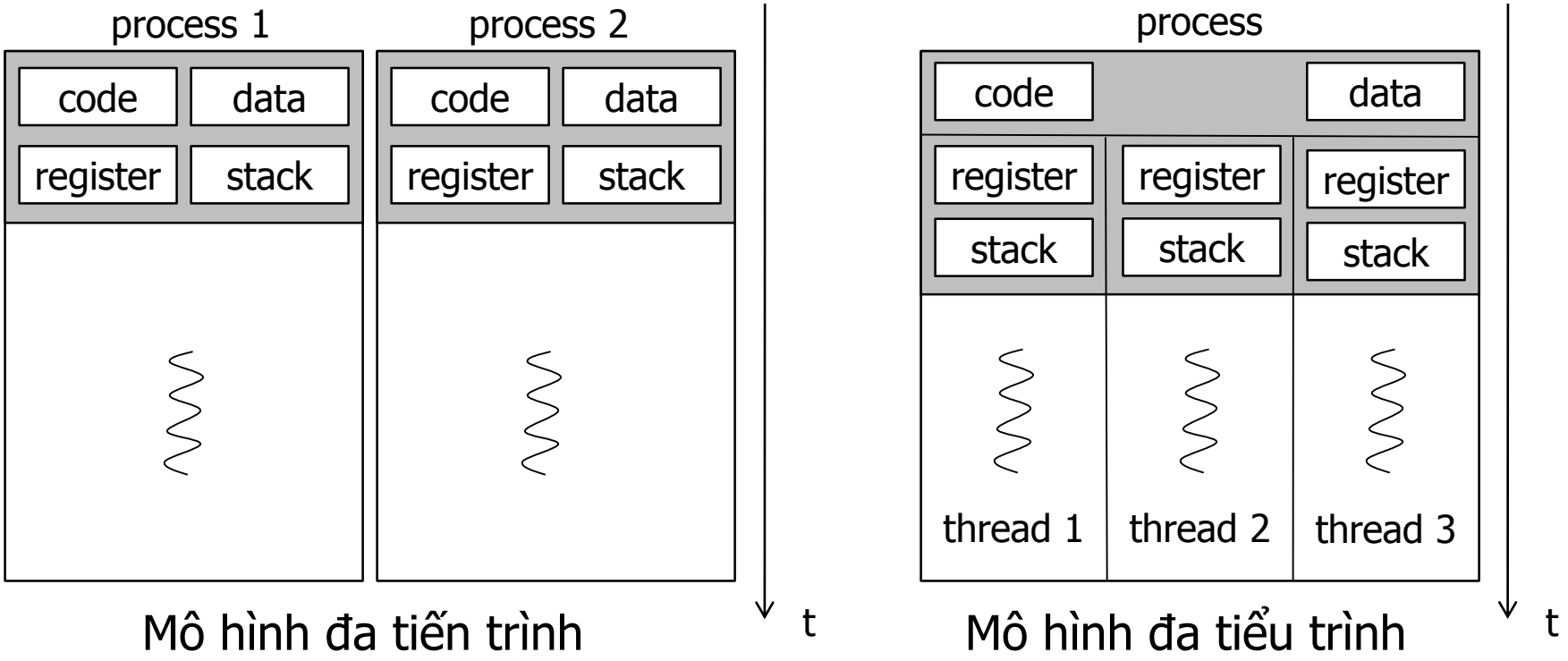
1. Khái niệm tiểu trình

- Một tiến trình thì chỉ có thể thực hiện một công việc tại một thời điểm.
- Nhu cầu đòi hỏi một tiến trình phải có thể thực hiện nhiều việc cùng lúc. Ví dụ:
  - Tiến trình của Web Server có thể phục vụ nhiều client
  - Chương trình chat cho phép chat với nhiều người.

→ Mô hình *đa tiểu trình*: một tiến trình thực hiện song song cùng lúc nhiều công việc, mỗi công việc được gọi là 1 tiểu trình.


nson_222
Conversation Edit View Actions Help
Call
IMVironment
Activities
Photos
nson_222
Add User
Ignore User
Send
snam_333
Conversation Edit View Actions Help
Call
IMVironment
Activities
Photos
snam_333
Add User
Ignore User
You currently appear offline to snam_333.
Send
thanhle_123
Conversation Edit View Actions Help
Call
IMVironment
Activities
Photos
thanhle_123
Add User
Ignore User
Send
dtam_555
Conversation Edit View Actions Help
Call
IMVironment
Activities
Photos
dtam_555
Add User
Ignore User
You currently appear offline to dtam_555.
Send
YAHOO! MESSENGE...
Messenger Contacts Actions Help
Add a Contact
Plug-ins
NEW YAHOO! MESSENGER KEEPS YOU CONNECTED.
REPLAY
Get Started
YAHOO! WEB SEARCH
YAHOO! MESSENGER
Messenger Contacts Actions Help
Add a Contact
Plug-ins
PARTY CRASHERS
SOULJA BOY TELL 'EM
PEPSI MUSIC ON YAHOO!
Take Me There
YAHOO! WEB SEARCH

Mô hình đa tiểu trình



Mọi tiến trình mặc định là có 1 tiểu trình thi hành lệnh (hàm main).

Ví dụ về đa tiểu trình:

Chương trình thực hiện song song 2 công việc "HelloWorld" và "HelloUniverse", in mỗi dòng chữ ra màn hình 1000 lần

```
using System;
using System.Threading;
class Program
{
    static void HelloWorld()
    {
        for (int i = 0; i < 1000; i++) Console.Write(" World  ");
    }

    static void HelloEarth()
    {
        for (int i = 0; i < 1000; i++) Console.Write("  Earth ");
    }

    static void Main(string[] args)
    {
        ThreadStart ts1 = new ThreadStart(HelloWorld);
        Thread thread1 = new Thread(ts1);
        ThreadStart ts2 = new ThreadStart(HelloEarth);
        Thread thread2 = new Thread(ts2);
        thread1.Start();
        thread2.Start();
        Console.ReadKey();
    }
}
```

2. Phân biệt tiến trình và tiểu trình

- Các thông tin trong PCB tiến trình:
  - CPU – con trỏ lệnh và các thanh ghi.
  - Memory: Code, Data, Files
  - Memory: Stack
- Các thông tin trong TCB tiểu trình:
  - CPU – con trỏ lệnh và các thanh ghi
  - Memory: Stack

3. Ưu điểm của tiểu trình so với tiến trình

- Đáp ứng nhu cầu thực hiện nhiều công việc cùng lúc của 1 chương trình.
- Chia sẻ tài nguyên: Dễ dàng chia sẻ tài nguyên giữa các tiểu trình (vì cùng thuộc một tiến trình nên có thể sử dụng các biến của tiến trình)
- Quản lý tiểu trình ít tốn kém hơn so với quản lý tiến trình
- Tận dụng được lợi ích của các kiến trúc đa bộ xử lý MultiProcessor, MultiCore

4. Điều phối tiểu trình của Windows

- Các tiểu trình trong một tiến trình cũng có độ ưu tiên riêng:
  - TIME_CRITICAL (+15)
  - HIGHEST (+2)
  - ABOVE_NORMAL (+1)
  - NORMAL
  - BELOW_NORMAL (-1)
  - LOWEST (-2)
  - IDLE (-15)

- Giá trị của các độ ưu tiên này phụ thuộc vào độ ưu tiên của tiến trình.

Q & A

Câu hỏi ôn tập

1. Nêu khái niệm tiến trình. CPU burst là gì? I/O burst là gì?
2. Tại sao cần đa tiến trình? CPU thực hiện đa tiến trình như thế nào?
3. Nêu 5 trạng thái tiến trình, và vẽ sơ đồ chuyển trạng thái.
4. Cho biết một số thông tin mà PCB lưu giữ. Mục đích của PCB để làm gì trong quá trình Context Switch?
5. Context Switch do cái gì thực hiện? Nêu các bước thực hiện context switch.
6. Nêu sự khác nhau giữa điều phối độc quyền khác điều phối không độc quyền.

7. Hãy nêu mức độ ưu tiên của các chiến lược FCFS, Round-Robin, SJF, Priority-based, Multilevel Feedback đối với những tiến trình có CPU burst ngắn.
8. Trong chiến lược Round Robin, nếu chọn time quantum quá nhỏ thì ảnh hưởng gì đến sự hoạt động của hệ thống? Còn nếu chọn time quantum quá lớn?
9. Phân biệt sự khác nhau giữa tiến trình và tiểu trình. Tại sao bên cạnh khái niệm tiến trình cần có khái niệm tiểu trình?

Bài tập

1. Xét tập các tiến trình sau:

Tiến trình	*Thời điểm vào RL*	*Thời gian CPU*	*Độ ưu tiên*
P_1	0	8	3
P_2	1	3	1
P_3	2.5	2	5
P_4	3	1	4
P_5	4.5	5	2

Cho biết kết quả điều phối theo chiến lược FCFS. Tính thời gian chờ cho từng tiến trình và thời gian chờ trung bình

2. Làm tương tự bài tập 1 cho các chiến lược
   - SJF
   - Round-Robin với time quantum q= 2
   - Độ ưu tiên độc quyền (độ ưu tiên 1>2>3>…)
   - Độ ưu tiên không độc quyền
3. Cho các tiến trình sau:

Tiến trình	*Thời điểm vào RL*	*Thời gian CPU*
P_1	0	8
P_2	0.4	4
P_3	1.0	1

Thực hiện tương tự bài tập 1 cho chiến lược FCFS và SJF

Bài Tập 4

Process	Burst Time
P1	10
P2	29
P3	3
P4	7
P5	12

Tất cả đều đến ở thời điểm 0

a)Xét các giải thuật FCFS, SJF, và RR với quantum time = 10

Giải thuật nào cho:

b)Thời gian đợi trung bình nhỏ nhất?

c)Thời gian đáp ứng trung bình tốt nhất

d)Thời gian hoàn thành trung bình của process nhỏ nhất?

Bài tập 5

- Round Robin. Tính thời gian đợi trung bình và thời gian hoàn thành trung bình

Process	Burst Time
P1	10
P2	29
P3	3
P4	7
P5	12

  - Quantum time = 10
  - Quantum time = 5

Bài tập 6

- Xét tập các tiến trình sau (với thời gian yêu cầu CPU và độ ưu tiên kèm theo) :

Tiến trình	Thời điểm vào RL	Thời gian CPU	Độ ưu tiên
P_1	0	10	3
P_2	1	3	2
P_3	2	2	1
P_4	3	1	2
P_5	4	5	4

- Vẽ giản đồ Gautt và tính thời gian đợi trung bình và thời gian lưu lại trong hệ thống trung bình (turnaround time) cho các giải thuật?
  - a) ***SJF***
  - ***b) RR*** với quantum time = 2,
  - ***c) Đ***iều phối theo độ ưu tiên độc quyền (độ ưu tiên 1 > 2 > ...)

Bài tập 7

Xét 1 tập các process sau có thời gian thực thi CPU tính bằng mili giây:

Process	Burst - time	Priority
P1	8	3
P2	2	1
P3	4	3
P4	2	4
p5	5	2

Giả sử thứ tự đến để thực thi của các process là P1, P2, P3, P4, P5. (tất cả process này đều đến tại thời điểm bằng 0).

1. Vẽ sơ đồ Gautt thực thi của các process theo giải thuật định thời: FCFS, SJF, RR (quantumn = 1), Priority (càng nhỏ càng ưu tiên, cùng độ ưu tiên xét Pi và Pj, Pi ưu tiên hơn nếu i < j).
2. Thời gian đợi của các process trong từng giải thuật định thời?

Bài tập 7 (tt)

3. Tính thời gian hoàn thành(turnaround time) của từng process cho từng giải thuật?

Trong các giải thuật sau, giải thuật nào có thể gây ra trường hợp 1 process có thể không bao giờ được thực thi:

a) First come, first serve
b) Shortest job first
c) Round robin
d) Priority.

Giải thích lý do tại sao xảy ra trường hợp trên?

Bài tập thực hành

1. Tìm hiểu chương trình Process Explorer
2. Tìm hiểu cơ chế điều phối tiến trình của Windows XP. Thực hiện suspend, resume, thay đổi độ ưu tiên tiến trình bằng Process Explorer. Xem tác động của việc thay đổi độ ưu tiên tiến trình
3. Viết chương trình C# cho phép thực hiện một chương trình trên đĩa, đồng thời theo dõi sự tồn tại của tiến trình.
4. Viết chương trình C# cho phép tắt 1 tiến trình

5. Viết chương trình C# liệt kê các tiến trình đang hoạt động trong hệ thống.
6. Viết chương trình đồng thời in ra 1000 từ “World” và 1000 từ “Earth”.
7. Viết chương trình thực hiện song song thuật 2 toán sắp xếp Selection Sort và Bubble Sort trên 2 mảng có cùng nội dung (mảng có kích thước n phần tử, các phần tử được sinh ngẫu nhiên)