Chương 3

QUẢN LÝ TIẾN TRÌNH

Nội dung chương 1

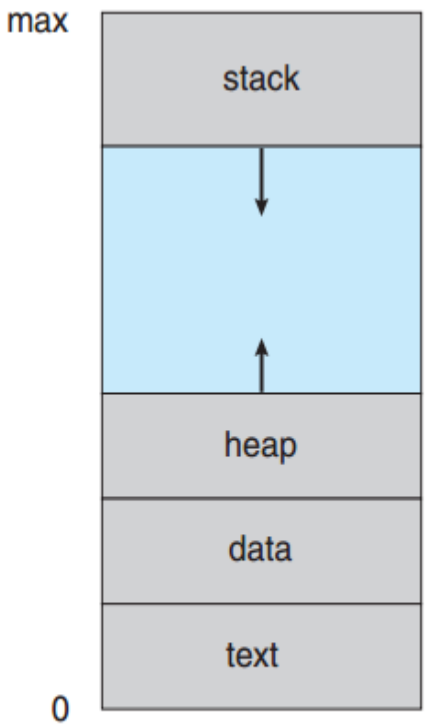
1. Khái niệm về tiến trình (process).
2. Tiểu trình (thread).
3. Điều phối tiến trình.
4. Đồng bộ tiến trình.
5. Tình trạng tắc nghẽn (deadlock)

Khái niệm về tiến trình (process)

- Tiến trình là một chương trình đang xử lý
- Mỗi tiến trình có một không gian địa chỉ, một con trỏ lệnh, một tập các thanh ghi và stack riêng.
- Tiến trình có thể cần đến một số tài nguyên như CPU, bộ nhớ chính, các tập tin và thiết bị nhập/xuất.
- Hệ điều hành sử dụng bộ điều phối (scheduler) để điều phối việc thực thi của các tiến trình.
- Trong hệ thống có những tiến trình của hệ điều hành và tiến trình của người dùng.
- Một tiến trình bao gồm Text section (program code), Data section (chứa global variables),

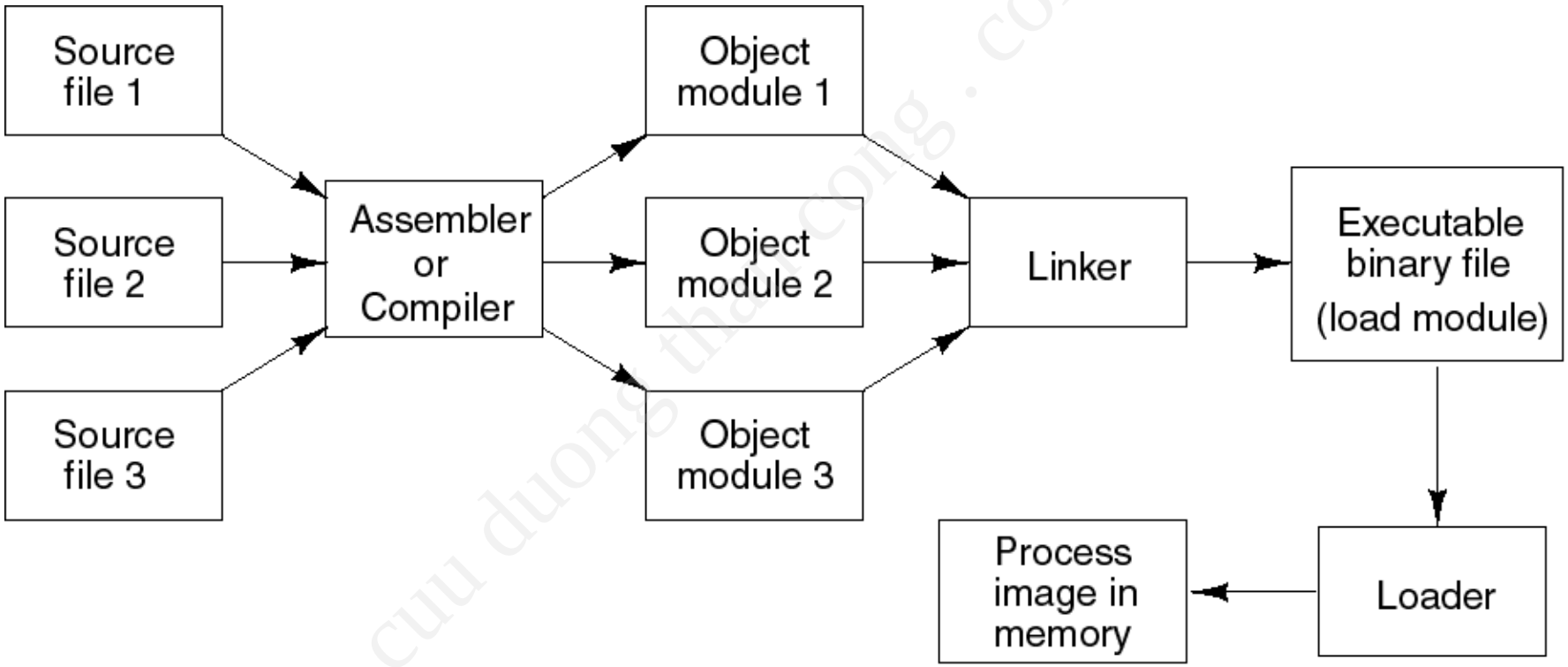
Khái niệm về tiến trình (process)

- Vùng code: chứa danh sách mã lệnh của CT
- Vùng static data: chứa các biến dữ liệu được khai báo tường minh trong CT.
- Vùng dynamic data: chứa các vùng nhớ dữ liệu được cấp phát động này biến động theo thời gian.
- Vùng stack: phục vụ cho việc gọi hàm trong chương trình. Kích thước vùng này biến động theo thời gian.



Tiến trình trong bộ nhớ

Khái niệm về tiến trình (process)



Các bước nạp chương trình vào bộ nhớ

Khái niệm về tiến trình (process)

Các bước hệ điều hành khởi tạo tiến trình

- ✓ Cấp phát một định danh duy nhất (process number hay process identifier, pid) cho quá trình
- ✓ Cấp phát không gian nhớ để nạp quá trình
- ✓ Khơi tạo khởi dữ liệu Process Control Block (PCB) cho quá trình

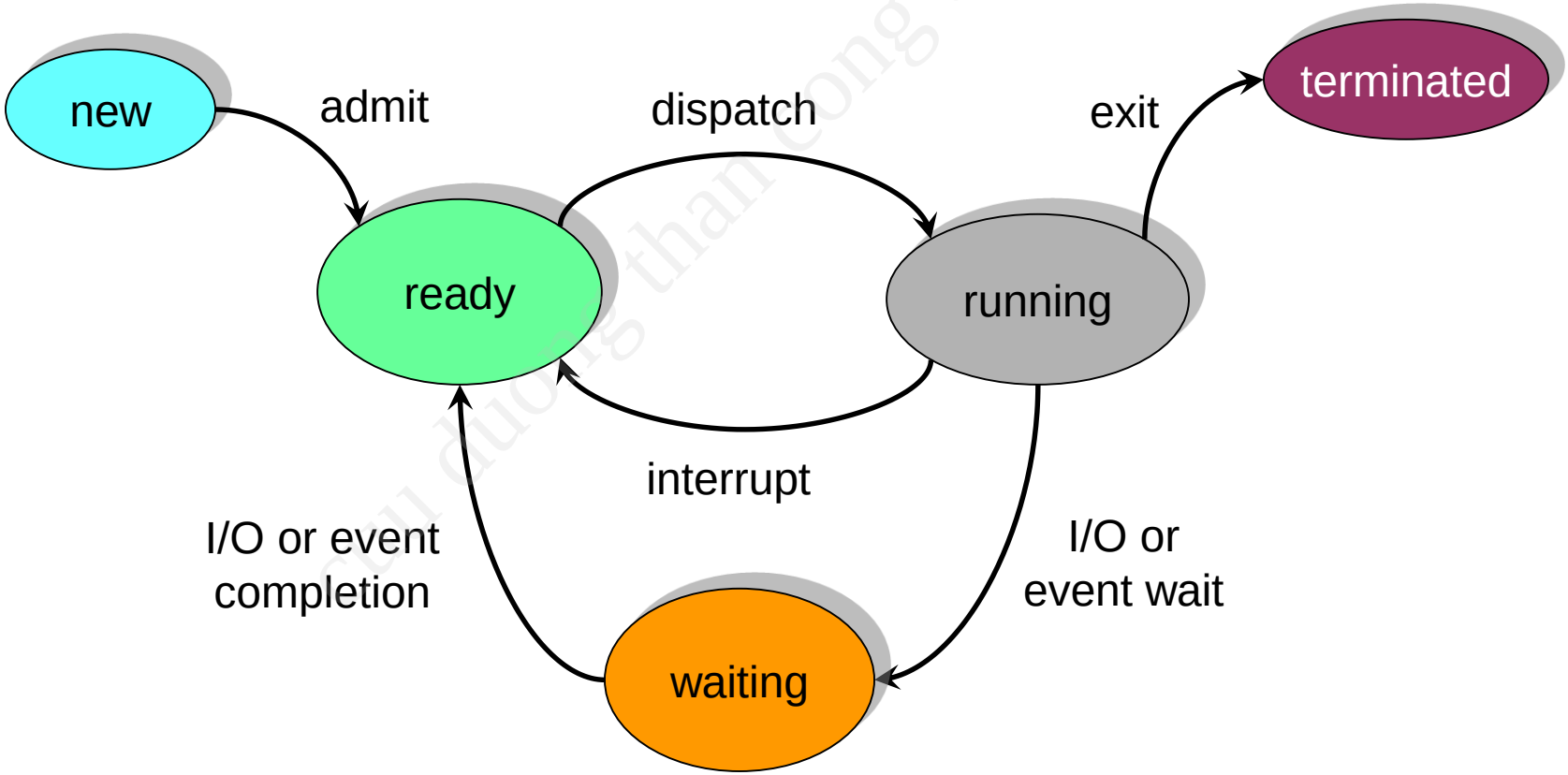
  PCB là nơi hệ điều hành lưu các thông tin về quá trình
- ✓ Thiết lập các mối liên hệ cần thiết (vd: sắp PCB vào hàng đợi định thời,…)

Khái niệm về tiến trình (process)

- ✓ *new:* tiến trình vừa được tạo
- ✓ *ready:* tiến trình đã có đủ tài nguyên, chỉ còn cần CPU
- ✓ *running:* các lệnh của tiến trình đang được thực thi
- ✓ *waiting:* hay là *blocked*, tiến trình đợi I/O hoàn tất, tín hiệu.
- ✓ *terminated:* tiến trình đã kết thúc.

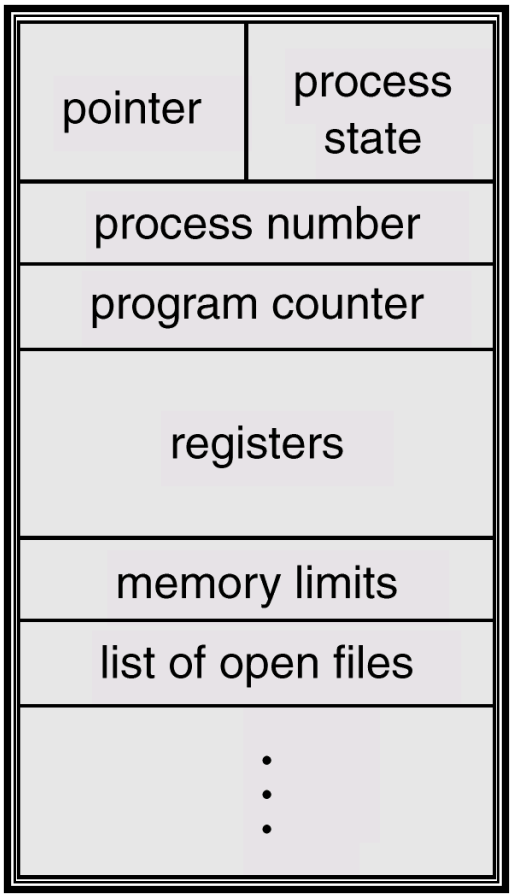
Khái niệm về tiến trình (process)

Chuyển đổi giữa các trạng thái của tiến trình



Khái niệm về tiến trình (process)

- Mỗi tiến trình trong hệ thống đều được cấp phát một *Process Control Block* (PCB)
- PCB là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành



Khái niệm về tiến trình (process)

Chuyển ngữ cảnh (Context switch)

- *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.
- *Chuyển 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.

Khái niệm về tiến trình (process)

Chuyển ngữ cảnh (Context switch)

- *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.
- *Chuyển 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.

Khái niệm về tiến trình (process)

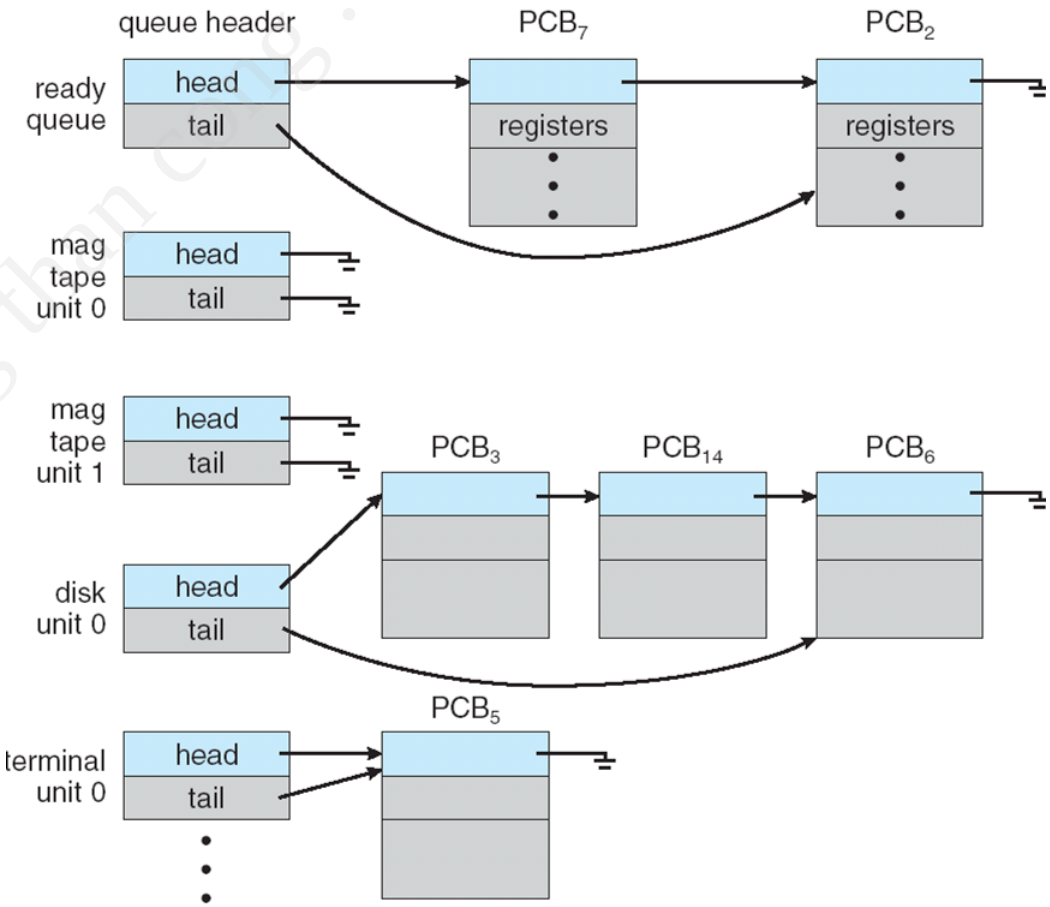
Định thời tiến trình

- Tại sao phải định thời?
  - Multiprogramming
    - ✓ Có nhiều tiến trình phải thực thi luân phiên nhau.
    - ✓ Mục tiêu: cực đại hiệu suất sử dụng của CPU.
  - Time-sharing
    - ✓ Cho phép users tương tác với tiến trình đang thực thi
    - ✓ Mục tiêu: tối thiểu thời gian đáp ứng
- Một số khái niệm cơ bản
  - Các *bộ định thời* (scheduler)
  - Các *hàng đợi định thời* (scheduling queue)

Khái niệm về tiến trình (process)

Các hàng đợi định thời

- Job queue
- Ready queue
- Device queues
- …



Khái niệm về tiến trình (process)

Thêm medium-term scheduling

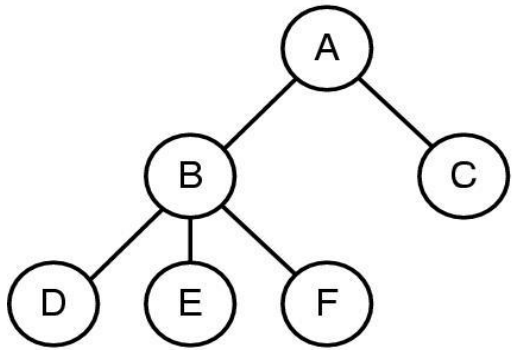
- Đôi khi hệ điều hành (như time-sharing system) có thêm medium-term scheduling để điều chỉnh mức độ multiprogramming của hệ thống
- *Medium-term scheduler*
  - ✓ Chuyển tiến trình từ bộ nhớ sang đĩa (swap out)
  - ✓ Chuyển tiến trình từ đĩa vào bộ nhớ (swap in)

Khái niệm về tiến trình (process)

- Tạo lập tiến trình (create)
- Kết thúc tiến trình (destroy)
- Tạm dừng tiến trình (suspend)
- Tái kích hoạt tiến trình (resume)
- Thay đổi độ ưu tiên tiến trình (change priority)

Khái niệm về tiến trình (process)

- Định danh cho tiến trình mới phát sinh.
- Đưa tiến trình vào danh sách quản lý của hệ thống.
- Xác định độ ưu tiên cho tiến trình.
- Tạo PCB cho tiến trình.
- Cấp phát các tài nguyên ban đầu cho tiến trình.



Khái niệm về tiến trình (process)

- Tạo tiến trình mới (process creation)
  - Một tiến trình có thể tạo tiến trình mới thông qua một system call (vd: hàm fork trong Unix)
    - ✓ Ví dụ: (Unix) Khi user đăng nhập hệ thống, một command interpreter (shell) sẽ được tạo ra cho user.
  - Tiến trình được tạo là tiến trình *con* của tiến trình tạo (tiến trình *cha*). Quan hệ cha-con định nghĩa một *cây tiến trình.*

Khái niệm về tiến trình (process)

- Tạo tiến trình mới
  - Chia sẻ tài nguyên của tiến trình cha
    - ✓ Tiến trình cha và con chia sẻ mọi tài nguyên
    - ✓ Tiến trình con chia sẻ một phần tài nguyên của cha
  - Trình tự thực thi
    - ✓ Tiến trình cha và con thực thi đồng thời (concurrently)
    - ✓ Tiến trình cha đợi đến khi các tiến trình con kết thúc.

Khái niệm về tiến trình (process)

- Kết thúc tiến trình
  - Tiến trình tự kết thúc
    - ✓ Tiến trình kết thúc khi thực thi lệnh cuối và gọi system routine exit.
  - Tiến trình kết thúc do tiến trình khác (có đủ quyền, vd: tiến trình cha của nó)
    - ✓ Gọi system routine abort với tham số là pid (process identifier) của tiến trình cần được kết thúc.
  - Hệ điều hành thu hồi tất cả các tài nguyên của tiến trình kết thúc (vùng nhớ, I/O buffer,…)

Khái niệm về tiến trình (process)

Cộng tác giữa các tiến trình

- Trong quá trình thực thi, các tiến trình có thể *cộng tác* (cooperate) để hoàn thành công việc.
- Các tiến trình cộng tác để:
  - Chia sẻ dữ liệu (information sharing).
  - Tăng tốc tính toán (computational speedup)
    - ✓ Nếu hệ thống có nhiều CPU, chia công việc tính toán thành nhiều công việc tính toán như chạy song song
  - Thực hiện một công việc chung.
    - ✓ Xây dựng một phần mềm phức tạp bằng cách chia thành các module/process hợp tác nhau.
- Sự cộng tác giữa các tiến trình yêu cầu hệ điều hành hỗ trợ cơ chế giao tiếp và cơ chế đồng bộ hoạt động của các tiến trình.

Tiểu trình (thread)

- Khái niệm tiến trình truyền thống: tiến trình gồm:
  - Không gian địa chỉ (text section, data section)
  - Một luồng thực thi duy nhất (single thread of execution):
    - ✓ program counter
    - ✓ các register
    - ✓ stack
  - Các tài nguyên khác (các open file, các quá trình con,…)

Tiểu trình (thread)

- Mở rộng khái niệm tiến trình truyền thống bằng cách hiện thực nhiều luồng thực thi trong cùng một môi trường của tiến trình.
- Khi đó tiến trình gồm:
  - Không gian địa chỉ (text section, data section)
  - Một hay nhiều luồng thực thi (thread of execution), mỗi luồng thực thi (thread) có riêng
    - ✓ program counter
    - ✓ các register
    - ✓ stack
  - Các tài nguyên khác (các open file, các quá trình con,…)

Tiểu trình (thread)

- Mô hình đa luồng:
  - Các thread trong cùng một process chia sẻ code section, data section và tài nguyên khác (các file đang mở,...) của process.
  - Tiến trình đa luồng (multithreaded process) là tiến trình có nhiều luồng.

Tiểu trình (thread)

❖ Mô hình đa luồng:

Ưu điểm

- ➢ Tính đáp ứng (responsiveness) cao cho các ứng dụng tương tác multithreaded
- ➢ Chia sẻ tài nguyên (resource sharing): vd memory
- ➢ Tiết kiệm chi phí hệ thống (lợi ích kinh tế)
  - ✓ Chi phí tạo/quản lý thread nhỏ hơn so với quá trình
  - ✓ Chi phí chuyển ngữ cảnh giữa các thread nhỏ hơn so với quá trình
- ➢ Tận dụng kiên trúc đa xử lý (multiprocessor)
  - ✓ Mỗi thread chạy trên một processor riêng, do đó tăng mức độ song song của chương trình.

Tiểu trình (thread)

User Thread

- Một thư viện thread (thread library, run-time system) được hiện thực trong user space để hỗ trợ các tác vụ lên thread
  - Thư viện thread cung cấp các hàm khởi tạo, định thời và quản lý thread như
    - ✓ thread_create
    - ✓ thread_exit
    - ✓ thread_wait
    - ✓ thread_yield
  - Thư viện thread dùng *Thread Control Block* (TCB) để lưu trạng thái của user thread (program counter, các register, stack)
- Kernel không biết sự có mặt của user thread

Tiểu trình (thread)

Kernel Thread

- Cơ chế multithreading được hệ điều hành trực tiếp hỗ trợ
  - Kernel quản lý cả process và các thread
  - Việc định thời CPU được kernel thực hiện trên thread
- Cơ chế multithreading được hỗ trợ bởi kernel
  - Khởi tạo và quản lý các thread chậm hơn
  - Tận dụng được lợi thế của kiến trúc multiprocessor
  - Thread bị blocked không kéo theo các thread khác bị blocked.
- Một số hệ thống multithreading (multitasking)
  - Windows
  - Solaris
  - Linux

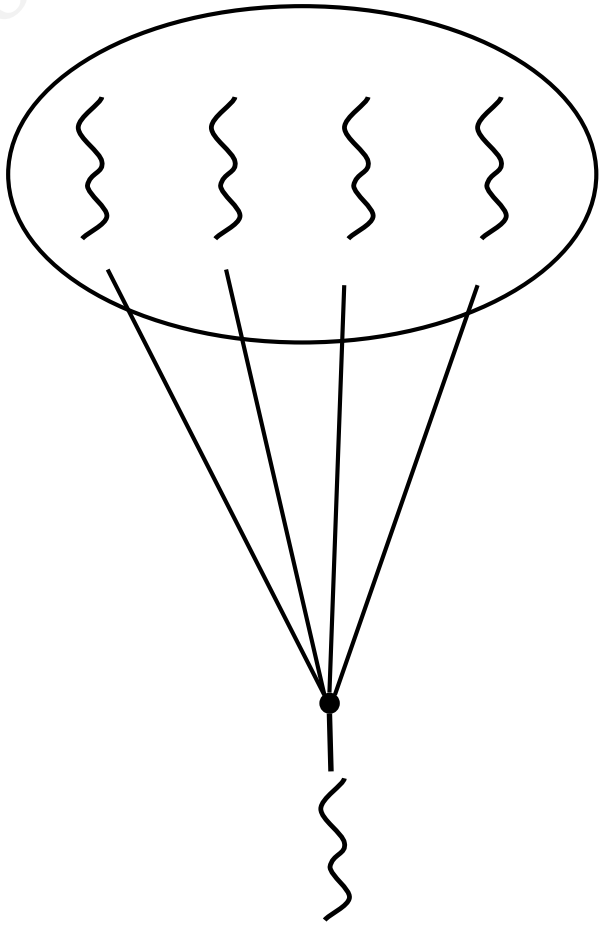
Tiểu trình (thread)

- Mộ số mô hình hiện thực thread
  - Mô hình *many-to-one*
  - Mô hình *one-to-one*
  - Mô hình *many-to-many*

Tiểu trình (thread)

Mô hình many – to – one

- Nhiều user-level thread "chia sẻ" một kernel thread để thực thi
  - Việc quản lý thread được thực hiện thông qua các hàm của một thread library được gọi ở user level.
  - Blocking problem: Khi một thread trở nên blocked thì kernel thread cũng trở nên blocked, do đó mỗi thread khác của process cũng sẽ trở nên blocked.
- Có thể được hiện thực đối với hầu hết các hệ điều hành.

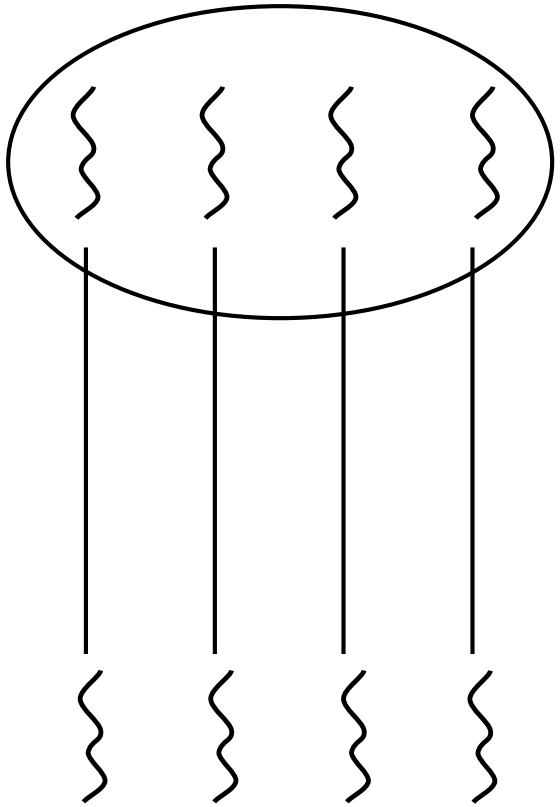


kernel thread

Tiểu trình (thread)

Mô hình one – to – one

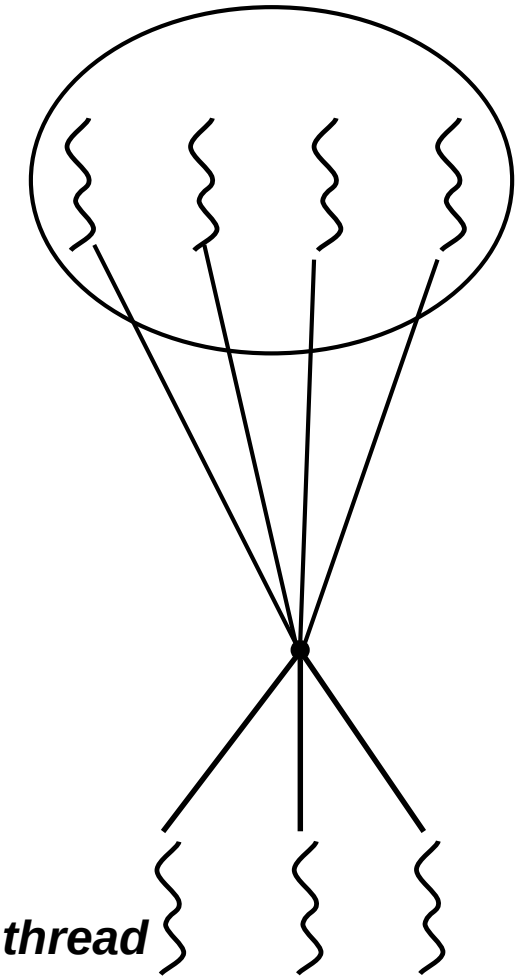
- Mỗi user-level thread thực thi thông qua một kernel thread riêng của nó
  - Mỗi khi một user thread được tạo ra thì cũng cần tạo một kernel thread tương ứng
- Hệ điều hành phải có cơ chế cung cấp được nhiều kernel thread cho một quá trình
- Ví dụ: Windows NT/2000



Tiểu trình (thread)

Mô hình many – to – many

- Nhiều user-level thread được phân chia thực thi (multiplexed) trên một số kernel thread.
  - Tránh được một số khuyết điểm cuûa hai mô hình many-to-one và one-to-one
- Ví dụ
  - Solaris 2
  - Windows NT/2000 với package ThreadFiber



Điều phối tiến trình

Một số khái niệm cơ bản

- Chu kỳ CPU-I/O: gồm chu kỳ thực thi CPU (CPU burst) và chu kỳ chờ đợi vào ra (I/O burst).
- *CPU-bound* process có thời gian sử dụng CPU nhiều hơn thời gian sử dụng I/O.
- *I/O-bound* process dùng phần lớn thời gian để đợi I/O.

Điều phối tiến trình

- Trong các hệ thống multitasking
  - Tại một thời điểm trong bộ nhớ có nhiều process
  - Tại mỗi thời điểm chỉ có một process được thực thi
  - Do đó, cần phải giải quyết vấn đề phân chia, lựa chọn process thực thi sao cho được hiệu quả nhất. Cần có chiến lược định thời CPU

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

Phân loại các hoạt động điều phối

- *Điều phối dài hạn* (long-term scheduling): xác định process nào được chấp nhận vào hệ thống
- *Điều phối trung hạn* (medium-term scheduling): xác định process nào được đưa vào (swap in), đưa ra khỏi (swap out) bộ nhớ chính
- *Điều phối ngắn hạn* (short-term scheduling): xác định process nào được thực thi tiếp theo

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

Điều phối dài hạn

- Xác định chương trình nào sẽ được đưa vào hệ thống để thực thi.
- Quyết định *độ-đa-lập-trình* (degree of multiprogramming).
- Nhiều process được đưa vào hệ thống:
  - Khả năng các process bị block sẽ giảm.
  - Sử dụng CPU hiệu quả hơn.
  - Mỗi process được phân chia khoảng thời gian sử dụng CPU thấp hơn.
- Không phân biệt process là CPU-bound hay I/O-bound

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

Điều phối trung hạn

- Quyết định việc đưa process vào bộ nhớ chính, hay ra khỏi bộ nhớ chính
- Phụ thuộc vào yêu cầu quản lý multiprogramming
  - Cho phép bộ định thời dài hạn chấp nhận nhiều process hơn số lượng process mà có tổng kích thước được chứa vừa trong bộ nhớ chính
  - Quá nhiều process thì sẽ làm tăng việc truy xuất đĩa, do đó cần phải lựa chọn độ-đa-lập-trình cho phù hợp
- Phần mềm quản lý bộ nhớ đảm nhiệm

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

Điều phối ngắn hạn

- Xác định process nào được thực thi tiếp theo, còn gọi là *Điều phối CPU*
- Được kích hoạt khi có một sự kiện có thể dẫn đến khả năng chọn một process để thực thi
  - Ngắt thời gian (clock interrupt)
  - Ngắt ngoại vi (I/O interrupt)
  - Lời gọi hệ thống (operating system call)
  - Signal

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

- Độ lợi CPU (CPU utilization)
  - Khoảng thời gian CPU bận, từ 0% đến 100%
  - Cần giữ cho CPU càng bận càng tốt
- Thời gian chờ (waiting time)
  - Thời gian chờ trong hàng đợi ready
  - Các process nên được chia sẻ việc sử dụng CPU một cách công bằng (fair share)
- Thông năng (throughput)
  - Số lượng process hoàn tất trong một đơn vị thời gian.

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

- Thời gian đáp ứng (response time)
  - Thời gian từ lúc có yêu cầu của người dùng (user request) đến khi có đáp ứng đầu tiên.
  - Thường là vấn đề với các I/O-bound process.
- Thời gian quay vòng (turnaround time)
  - Thời gian để một process hoàn tất, kể từ lúc vào hệ thống (submission) đến lúc kết thúc (termination).
  - Là một trị đặc trưng cần quan tâm với các process thuộc dạng CPU-bound.
- Thời gian quay vòng trung bình (average turnaround time)

Điều phối tiến trình

Các tiêu chí điều phối tiến trình

- Hướng đến người sử dụng (user-oriented)
  - Thời gian quay vòng (turnaround time)
    - ✓ Thời gian từ lúc nạp process đến lúc process kết thúc
    - ✓ Quan tâm với các hệ thống xử lý bó (batch system)
  - Thời gian đáp ứng (response time)
    - ✓ Quan tâm các hệ thống giao tiếp (interactive system)
- Hướng đến hệ thống (system-oriented)
  - Độ lợi CPU (CPU utilization)
  - Công bằng (fairness)
  - Thông năng (throughput): số process hoàn tất trong một đơn vị thời gian

Điều phối tiến trình

Hai thành phần của chiến lược định thời

- *Chế độ quyết định* (decision mode)
  - Chọn thời điểm hàm lựa chọn định thời thực thi.
  - *Nonpreemptive – Không độc quyền*
    - ✓ Một process sẽ ở trạng thái running cho đến khi nó bị block hoặc nó kết thúc.
  - *Preemptive – Độc quyền*
    - ✓ Process đang thực thi có thể bị ngắt và chuyển về trạng thái ready.
    - ✓ Tránh trường hợp một process độc chiếm (monopolizing) CPU.

Điều phối tiến trình

Hai thành phần của chiến lược định thời

- Hàm định thời có thể được thực thi khi có quá trình
  - ✓ Chuyển từ trạng thái running sang waiting (1)
  - ✓ Chuyển từ trạng thái running sang ready (2)
  - ✓ Chuyển từ trạng thái waiting, new sang ready (3)
  - ✓ Kết thúc thực thi (4)
- Định thời *nonpreemptive*: chỉ thực thi hàm định thời trong trường hợp 1 và 4
- Định thời *preemptive*: ngoài trường hợp 1 và 4 còn thực thi thêm hàm định thời trong trường hợp 2 hoặc 3 (hoặc cả hai)

Điều phối tiến trình

Hai thành phần của chiến lược định thời

- Hàm định thời có thể được thực thi khi có quá trình
  - ✓ Chuyển từ trạng thái running sang waiting (1)
  - ✓ Chuyển từ trạng thái running sang ready (2)
  - ✓ Chuyển từ trạng thái waiting, new sang ready (3)
  - ✓ Kết thúc thực thi (4)
- Định thời *nonpreemptive*: chỉ thực thi hàm định thời trong trường hợp 1 và 4
- Định thời *preemptive*: ngoài trường hợp 1 và 4 còn thực thi thêm hàm định thời trong trường hợp 2 hoặc 3 (hoặc cả hai)

Điều phối tiến trình

Dispatcher

- Chuyển quyền điều khiển CPU về cho process được chọn bởi bộ định thời ngắn hạn:
  - Chuyển ngữ cảnh (sử dụng thông tin ngữ cảnh trong PCB)
  - Chuyển về user mode
  - Nhảy đến vị trí thích hợp trong chương trình ứng dụng để khởi động lại chương trình (chính là program counter trong PCB)
- Công việc này gây ra phí tổn
  - *Dispatch latency*: thời gian mà dispatcher dừng một process và khởi động một process khác

Điều phối tiến trình

Các giải thuật điều phối tiến trình

First Come First Served (FCFS)

- Hàm lựa chọn: chọn process đợi trong hàng đợi ready lâu nhất
- Chế độ quyết định: nonpreemptive
  - Một process sẽ được thực thi cho đến khi nó bị block hoặc kết thúc
- FCFS thường được quản lý bằng một FIFO queue

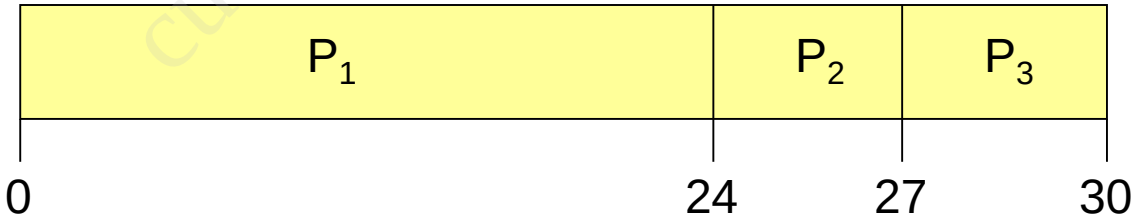
Điều phối tiến trình

Các giải thuật điều phối tiến trình

First Come First Served (FCFS)

Process	Burst time (ms)
P1	24
P2	3
P3	3

- Giả sử các process đến theo thứ tự *P1*, *P2*, *P3*
- *Giản đồ Gantt* cho việc định thời là:



Thời gian đợi cho *P1* = 0, *P2* = 24, *P3* = 27

- Thời gian đợi trung bình: (0 + 24 + 27)/3 = 17

Điều phối tiến trình

Các giải thuật điều phối tiến trình

First Come First Served (FCFS)

- FCFS không công bằng với các process có CPU burst ngắn. Các process này phải chờ trong thời gian dài (so với thời gian mà nó cần phục vụ) mới được sử dụng CPU, hay FCFS "ưu tiên" các process thuộc dạng CPU bound.
- FCFS có giải quyết để tránh trường hợp *trì hoãn vô hạn định* (starvation hay indefinite blocking)?
- FCFS thường được sử dụng trong các hệ thống bó (batch system)

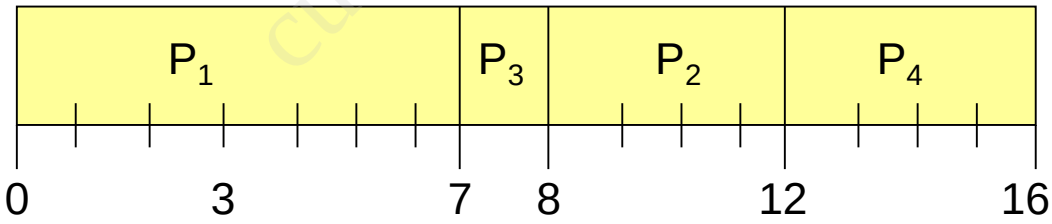
Điều phối tiến trình

Các giải thuật điều phối tiến trình

Shortest Job First (SJF)

Process	Thời điểm đến	Burst time (ms)
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Giản đồ Gantt khi định thời theo SJF



- Thời gian đợi trung bình = (0 + 6 + 3 + 7)/4 = 4

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Shortest Job First (SJF)

- Tương ứng với mỗi process cần có độ dài của CPU burst tiếp theo
- Hàm lựa chọn: chọn process có độ dài CPU burst nhỏ nhất
- Chứng minh được: SJF tối ưu trong việc giảm thời gian đợi trung bình
- Nhược điểm: Cần phải ước lượng thời gian cần CPU tiếp theo của process

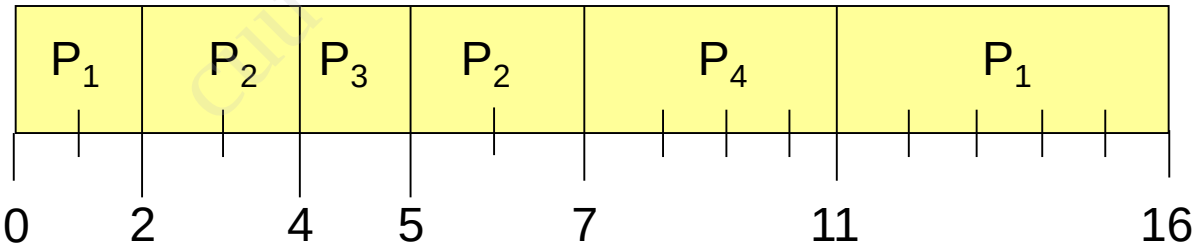
Điều phối tiến trình

Các giải thuật điều phối tiến trình

Shortest Remaining Time First (SRTF)

Process	Thời điểm đến	Burst time (ms)
P1	0	7
P2	2	4
P3	4	1
P4	5	4

- Giản đồ Gantt khi định thời theo SRTF



- Thời gian đợi trung bình = (9 + 1 + 0 + 2)/4 = 3
  - Tốt hơn giải thuật nonpreemptive SJF

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Shortest Remaining Time First (SRTF)

- Tránh trường hợp độc chiếm CPU của các process có thời gian thực thi dài
- Cần phải quản lý thời gian thực thi còn lại của các process
- Có thời gian quay vòng tốt hơn SJF
- Process có thời gian thực thi ngắn có độ ưu tiên cao

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Priority Scheduling

- Mỗi process sẽ được gán một độ ưu tiên
- CPU sẽ được cấp cho process có độ ưu tiên cao nhất
- Định thời sử dụng độ ưu tiên có thể:
  - Preemptive
  - Nonpreemptive

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Priority Scheduling

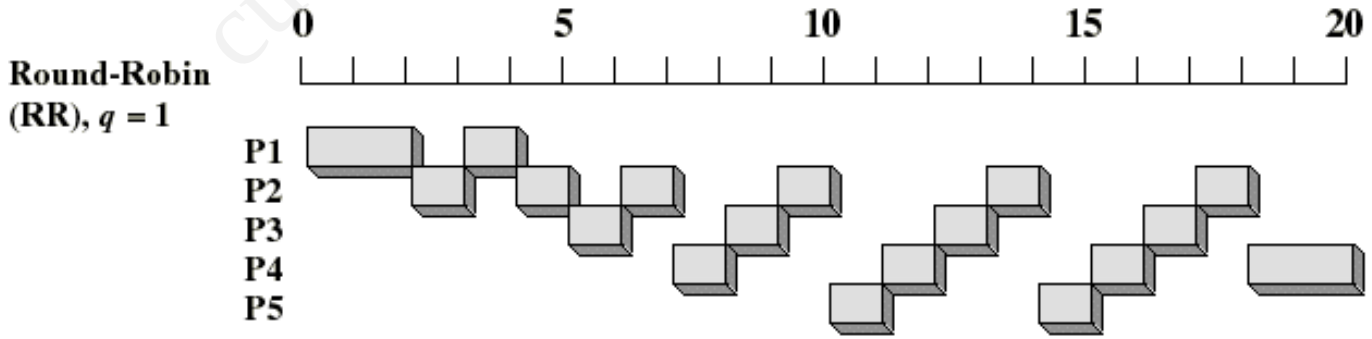
- SJF là một giải thuật định thời sử dụng độ ưu tiên với độ ưu tiên là thời gian sử dụng CPU dự-đoán
- Gán độ ưu tiên còn dựa vào:
  - Yêu cầu về bộ nhớ
  - Số lượng file được mở
  - Tỉ lệ thời gian dùng cho I/O trên thời gian sử dụng CPU.
- Vấn đề: trì hoãn vô hạn định – process có độ ưu tiên thấp có thể không bao giờ được thực thi
- Giải pháp: *aging* – độ ưu tiên của process sẽ tăng theo thời gian

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Round Robin (RR)

- Chế độ quyết định: preemptive
  - Khoảng thời gian tối đa cho phép (thường 10 - 100 ms) được đảm bảo bằng việc sử dụng timer interrupt
  - Process đang chạy hết thời gian sẽ được chuyển về cuối của hàng đợi ready



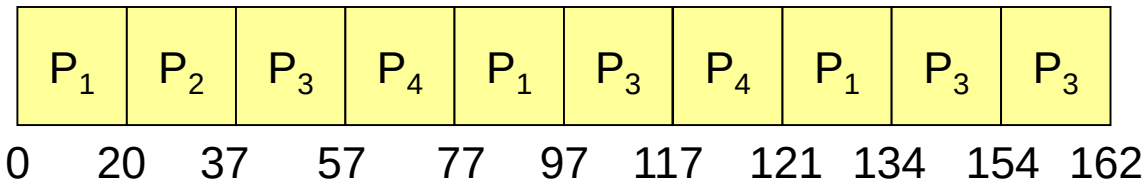
Điều phối tiến trình

Các giải thuật điều phối tiến trình

Round Robin (RR)

Process	Burst time (ms)
P1	53
P2	17
P3	68
P4	24

- Thời gian lượng định = 20 ms
- Giản đồ Gantt:

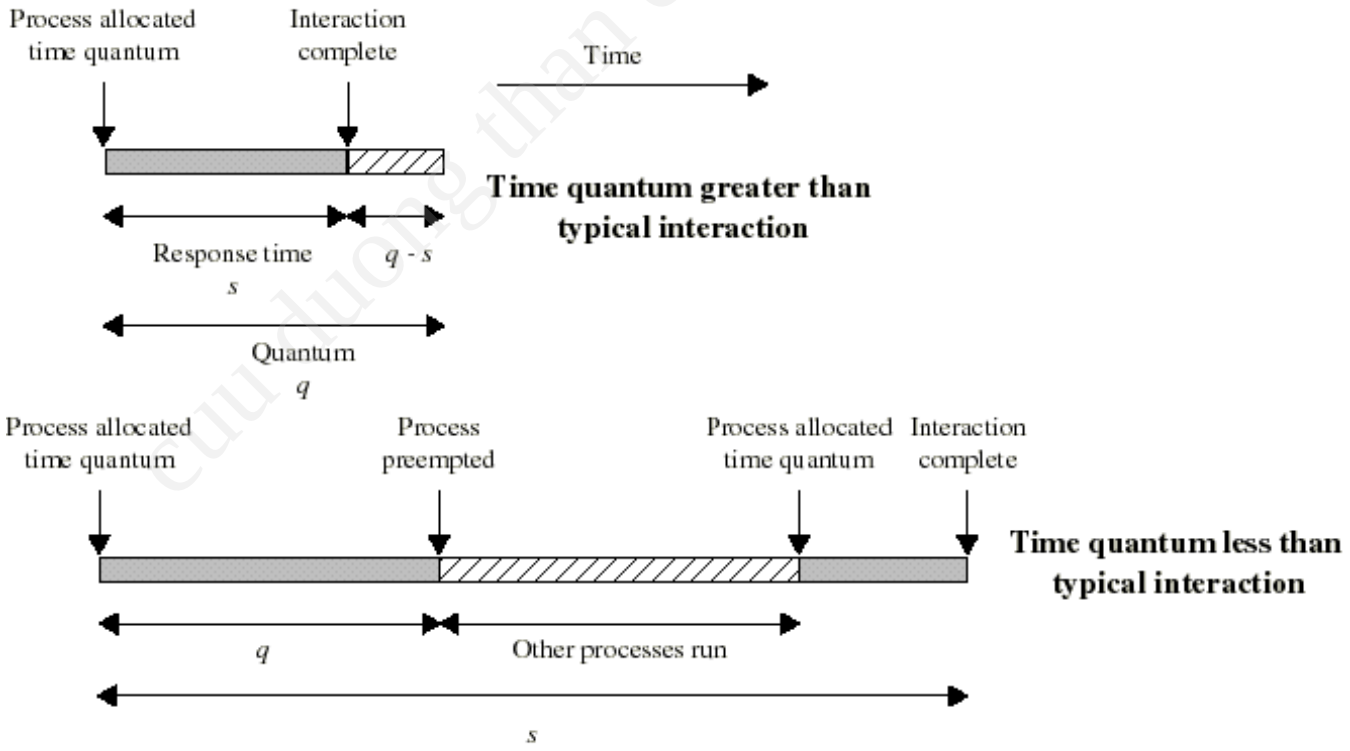


Thường có thời gian quay vòng cao hơn SJF, nhưng lại có thời gian đáp ứng tốt hơn

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Lượng định thời gian (time quantum) cho Round Robin



Điều phối tiến trình

Các giải thuật điều phối tiến trình

Round Robin

- Khi thực hiện chuyển process thì OS sẽ sử dụng CPU chứ không phải process của người dùng (*OS overhead*):
  - Dừng thực thi, lưu tất cả thông tin, nạp thông tin của process sắp thực thi
- Hiệu năng tùy thuộc vào kích thước của thời gian lượng định (còn gọi là time slice). Hàm phụ thuộc này không đơn giản
- Time slice ngắn thì đáp ứng nhanh
  - Có nhiều chuyển ngữ cảnh: Phí tổn sẽ cao.
- Time slice dài hơn thì throughput tốt hơn (do giảm phí tổn OS overhead) nhưng thời gian đáp ứng lớn
  - Nếu time slice quá lớn, RR trở thành FCFS.

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Round Robin

- Quantum time và thời gian chuyển process:
  - Nếu quantum time = 20 ms và thời gian chuyển process = 5 ms, thì phí tổn OS overhead chiếm 5/25 = 20%
  - Nếu quantum = 500 ms, thì phí tổn chỉ còn ≈ 1%
    - ✓ Nếu có nhiều người sử dụng trên hệ thống và thuộc loại interactive thì sẽ thấy đáp ứng rất chậm
  - Tùy tập công việc mà chọn quantum time phù hợp
  - Time slice nên lớn trong tương quan so sánh với thời gian chuyển process
  - Ví dụ với 4.3 BSD UNIX, time slice là 1 giây

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Round Robin

- Nếu có *n* process trong hàng đợi ready, và quantum time là *q*, thì mỗi process sẽ lấy 1/*n* thời gian CPU theo từng khối có kích thước lớn nhất là *q*
  - Sẽ không có process nào chờ lâu hơn (*n* - 1)*q* đơn vị thời gian
- RR sử dụng một giả thiết ngầm là tất cả các process đều có tầm quan trọng ngang nhau
  - RR không phù hợp với hệ thống xác định độ ưu tiên khác nhau cho mỗi process
- Các process dạng CPU-bound vẫn còn được "ưu tiên"

Điều phối tiến trình

Các giải thuật điều phối tiến trình

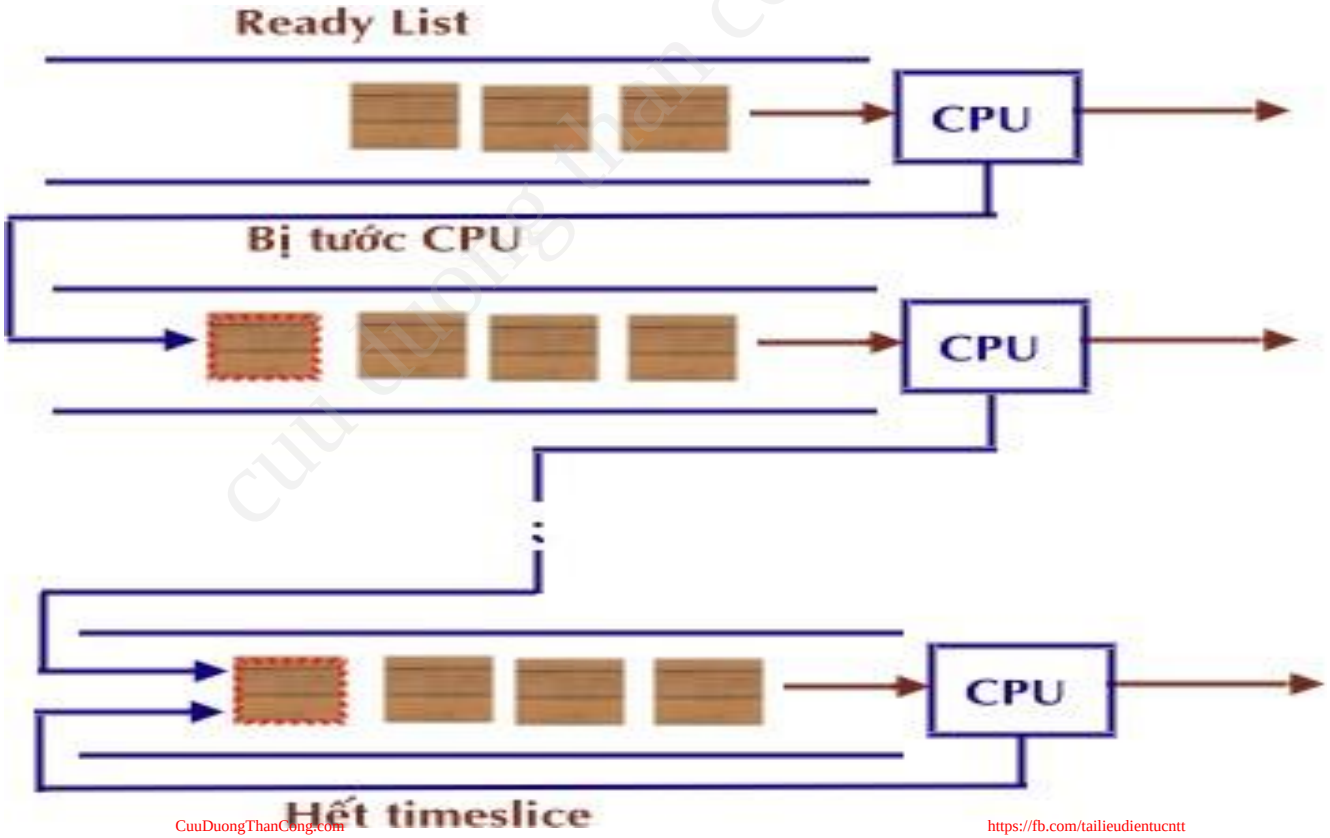
Thuật toán nhiều mức độ ưu tiên

- Danh sách sẵn sàng được chia thành nhiều danh sách.
- Mỗi danh sách gồm các tiến trình có cùng độ ưu tiên và được áp dụng một giải thuật điều phối riêng

Điều phối tiến trình

Các giải thuật điều phối tiến trình

Điều phối theo nhiều mức ưu tiên xoay vòng (Multilevel Feedback)



Điều phối tiến trình

Các giải thuật điều phối tiến trình

Chiến lược điều phối xổ số (Lottery)

- Mỗi tiến trình được cấp một "vé số".
- HĐH chọn 1 vé "trúng giải", tiến trình nào sỡ hữu vé này sẽ được nhận CPU.
- Là giải thuật độc quyền.
- Đơn giản, chi phí thấp, bảo đảm tính công bằng cho các tiến trình.