MÔN HỌC: HỆ ĐIỀU HÀNH
CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4

1. Tại sao phải định thời? Có những loại bộ định thời nào?

Mục tiêu của việc lập trình đa luồng là hướng đến việc luôn luôn phải có tiến trình sử dụng CPU, hay nói cách khác, là tối đa hoá việc sử dụng CPU. Ngoài ra, mục tiêu của các hệ thống time sharing (chia sẻ thời gian – tức các hệ thống HĐH hiện nay) là việc mang đến cho người dùng cảm giác chiếc máy tính của mình có thể làm được nhiều công việc cùng một lúc. Việc đó chỉ có thể đạt được thông qua việc chuyển quyền sử dụng CPU thật nhanh qua lại giữa các tiến trình.

Và để đạt được các mục tiêu nêu trên, trình định thời (Scheduler) sẽ lựa chọn trong các tiến trình hiện có để thực thi trên CPU. Nguyên nhân là do, trong một thời điểm nhất định, chỉ duy nhất có một tiến trình được quyền ở trạng thái running mà thôi.

Có 3 bộ định thời :

- Short-Term Scheduling (hay còn gọi là Dispatcher) : Dùng để định thời cho CPU.
    - Xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp.
    - Bộ định thời Short-Term sẽ được gọi mỗi khi có một trong các sự kiện/interrupt sau xảy ra :
        - Ngắt thời gian (clock interrupt).
        - Ngắt ngoại vi (I/O interrupt).
        - Lời gọi hệ thống (Operating System Call).
        - Signal.
- Medium-Term Scheduling : Dùng để định thời Swaping.
    - Process nào được đưa vào (swap-in), đưa ra khỏi (swap-out) bộ nhớ chính.
    - Được thực hiện bởi phần quản lý bộ nhớ và được thảo luận ở phần quản lý bộ nhớ.
- Long-Term Scheduling (hay còn gọi là Job Scheduler) :
    - Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi.
    - Điều khiển mức độ multiprogramming của hệ thống.
    - Long-Term Scheduling thường cố gắng duy trì xen lẫn CPU-Bound và I/O Bound Process.

2. Định thời CPU là gì? Bộ định thời nào chịu trách nhiệm thực hiện việc này?

   Định thời CPU là quá trình xác định tiến trình nào trong hàng đợi sẵn sàng (ready queue) sẽ được chiếm CPU để thực thi kế tiếp. Bộ định thời chịu trách nhiệm cho việc này là Short Term Scheduling.

3. Phí tổn gây ra khi định thời là gì?

   Phí tổn gây ra khi định thời, hay còn gọi là Dispatch Latency là thời gian mà bộ định thời dừng một tiến trình và khởi động một tiến trình khác.

4. Trình bày các tiêu chuẩn định thời CPU?

   Các thuật toán định thời CPU khác nhau có các tính chất khác nhau. Vì vậy, rất nhiều tiêu chí đã được đề ra để so sánh các thuật toán định thời CPU. Các tiêu chuẩn gồm:
   - Mức độ sử dụng CPU (CPU Utilization): Chúng ta muốn làm sao cho CPU càng bận càng tốt. Theo lý thuyết thì CPU utilization có thể đạt từ 0 đến 100 phần trăm. Trong hệ thống thực tế, thông thường CPU utilization sẽ đạt 40% và có thể đạt đến mức 90% cho hệ thống load nặng.
   - Thông lượng(throughput): Một trong những thước đo cho hiệu quả của quá trình làm việc đó chính là thông lượng. Thông lượng được tính bằng số lượng tiến trình đã được hoàn thành trong một đơn vị thời gian.
   - Thời gian hoàn thành (Turnaround time): Từ góc nhìn của một tiến trình nhất định, một số yếu tố quan trọng cần xem xét đó chính là khoảng thời gian cần thiết để thực thi tiến trình đó. Khoảng thời gian từ lúc tiến trình được ghi nhận đến khi hoàn thành chính là Turnaround Time. Vì vậy, Turnaround Time là tổng của tất cả các khoảng thời gian: Tiến trình trong hàng đợi, thực thi trên CPU và thực thi lệnh I/O.
   - Thời gian đợi (Waiting Time): Tổng thời gian tiến trình đã ở trong hàng đợi ready queue.
   - Thời gian đáp ứng (Response Time): thời gian từ lúc tiến trình xuất hiện cho đến khi thực hiện tiến trình đó lần đầu tiên.

5. Kể tên các giải thuật định thời CPU?

   - First Come, First Served (FCFS)
   - Shortest Job First (SJF)
   - Preemptive SJF (Shortest Remaining Time First – SRTF)
   - Priority Scheduling
   - Round-Robin (RR)
   - Highest Response Ratio Next
   - Multilevel Queue
   - Multilevel Feelback Queue

6. Mô tả và nêu ưu điểm, nhược điểm của từng giải thuật định thời sau: FCFS, SJF, SRTF, RR, Priority Scheduling, HRRN, MQ, MFQ.

FCFS :

- Mô tả :
  - Cơ chế thực thi :
    - Tiến trình nào yêu cầu CPU trước sẽ được cấp phát trước.
    - Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O.
  - Chế độ quyết định : Non-Preemptive.
  - Hiện thực : Sử dụng hàng đợi FIFO.
    - Tiến trình đi vào được thêm vào cuối hàng đợi.
    - Tiến trình được lựa chọn để xử lý được lấy từ đầu của queue.
- Ưu điểm :
  - Sẽ không bị starvation.
  - Thuật toán này dễ cài đặt. Code đơn giản.
- Nhược điểm :
  - Thời gian chờ trung bình của FCFS thường khá dài (VD : Một process có burst-time rất dài đến trước, khi đó các process có burst-time nhỏ sẽ phải chờ 1 khoảng thời gian rất lâu mới đến lượt thực thi).
  - Lãng phí thời gian do thời gian phần cứng trống khá nhiều (convoy effect).
  - Non-preemptive. Sẽ không hoạt động tốt trong các hệ thống chia sẻ thời gian (time-sharing system) khi các user đều mong muốn được sử dụng CPU trong một khoảng thời gian và không muốn delay quá lâu.

SJF :

- Mô tả :
  - Cơ chế thực thi :
    - Định thời công việc ngắn nhất trước (Burst-time nhỏ nhất).
    - Khi CPU được tự do, nó sẽ cấp phát cho tiến trình nào yêu cầu ít thời gian nhất để kết thúc (burst-time nhỏ nhất).
    - Burst-time có được từ việc dự đoán, dựa vào các lần chạy trước của tiến trình.
    - Nếu có 2 tiến trình cùng Burst-time, tiến trình nào vào hàng đợi trước sẽ được chạy trước (không xét độ ưu tiên).
  - Chế độ quyết định : Non-Preemptive.
- Ưu điểm : Tối ưu. Cho thời gian chờ đợi trung bình tối thiểu với một tập tiến trình cho trước.
- Nhược điểm :
  - Cần phải ước lượng thời gian cần CPU tiếp theo của process (Burst time).
  - Có thể xảy ra starvation nếu số lượng process có burst time nhỏ cần được thực thi quá nhiều.

SRTF :

- Mô tả :
  - Cơ chế thực thi :
    - (Tương tự SJF).

        - Nếu một tiến trình mới được đưa vào danh sách với chiều dài sử dụng CPU cho lần tiếp theo **nhỏ hơn** (lưu ý, chỉ **nhỏ hơn,** nếu burst-time bằng thì không preempt) thời gian còn lại của tiến trình đang xử lý, nó sẽ dừng hoạt động tiến trình hiện hành (**preempt**).
    - Chế độ quyết định : Preemptive.
- Ưu điểm :
    - Preemptive. Thời gian đáp ứng nhanh cho các tác vụ nhỏ.
    - Tránh việc một tác vụ lớn độc chiếm CPU.
    - Thời gian chờ đợi trung bình thường sẽ nhỏ hơn SJF.
- Nhược điểm :
    - (Các nhược điểm của SJF).
    - Tăng thời gian hoàn thành trung bình.

Priority Scheduling :

- Mô tả :
    - Cơ chế hoạt động :
        - Mỗi tiến trình sẽ được gán 1 độ ưu tiên.
        - CPU sẽ được cấp cho tiến trình có độ ưu tiên cao nhất.
        - Định thời sử dụng độ ưu tiên có thể là :
            - Preemptive : Khi một tiến trình mới xuất hiện có độ ưu tiên cao hơn, nó sẽ preempt tiến trình đang chạy.
            - Non-Preemptive : Tiến trình đang chạy sẽ tiếp tục chạy.
        - Nếu có 2 tiến trình cùng độ ưu tiên, thì tiến trình nào đến trước sẽ được chạy trước. Burst-time không được áp dụng để so sánh ở đây.
    - Chế độ quyết định : Non-Preemptive hoặc Preemptive.
- Ưu điểm :
    - Các tác vụ quan trọng sẽ được thực thi trước.
- Nhược điểm :
    - Có thể xảy ra starvation : Các 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).

Round Robin :

- Mô tả :
    - Cơ chế hoạt động :
        - Mỗi tiến trình nhận được một đơn vị nhỏ thời gian CPU (time-slice, quantum time), thông thường từ 10-100msec để thực thi.
        - CPU Schedulers sẽ chọn 1 tiến trình từ ready queue và “lên dây cót” một quantum cho tiến trình, sau đó cho tiến trình chạy. Lúc này, sẽ có 2 khả năng có thể xảy ra :
            - Thời gian chạy > Quantum : Khi đó, tiến trình sẽ bị interrupt và CPU Schedulers sẽ chọn tiếp tiến trình tiếp theo.
            - Thời gian chạy < Quantum : Tiến trình tiếp theo sẽ ngay lập tức được thực thi tiếp (không cần chờ hết quantum time của tiến trình trước), và tiến trình tiếp theo đó cũng được gán 1 quantum time.
        - Phụ thuộc nhiều vào quantum time :

            - Quantum time ngắn thì đáp ứng nhanh, tuy nhiên overhead lớn do chuyển ngữ cảnh nhiều. Quantum time phải > thời gian chuyển ngữ cảnh (context switch).
            - Quantum time dài thì đáp ứng chậm, tuy nhiên thông lượng (throughput) sẽ cao. Và khi quantum time quá lớn RR->FCFS (Quantum time lớn -> Không bao giờ bị ngắt -> Ai vào trước làm trước -> FCFS).
        - Khi cả tiến trình vừa thực thi xong và tiến trình mới cũng arrive vào cùng một thời điểm, thì tiến trình mới sẽ vào hàng đợi trước rồi mới đến tiến trình cũ.
        - Các tiến trình đều có độ ưu tiên giống nhau.
    - Chế độ quyết định : Preemptive.
- Ưu điểm :
    - Thời gian đáp ứng trung bình thường thấp -> Thích hợp cho các hệ thống time-sharing.
    - Không xảy ra tình trạng starvation.
- Nhược điểm :
    - Thời gian chờ đợi trung bình thường khá lớn.
    - Chuyển ngữ cảnh nhiều -> Hao phí cao.
    - Hiệu suất thuật toán phụ thuộc nhiều vào việc chọn quantum time.
    - Không thể sử dụng thuật toán nếu muốn các ứng dụng có độ ưu tiên khác nhau.

Highest Response Ratio Next (HRRN) :

- Mô tả :
    - Cơ chế hoạt động :
        - Chọn process tiếp có giá trị RR (Response Ratio) **lớn nhất**.
        - Các process ngắn được ưu tiên hơn vì service time (hay burst time) nhỏ.
    - Công thức :

$$response\ ratio = \frac{waiting\ time + estimated\ run\ time}{estimated\ run\ time} = 1 + \frac{waiting\ time}{estimated\ run\ time}$$

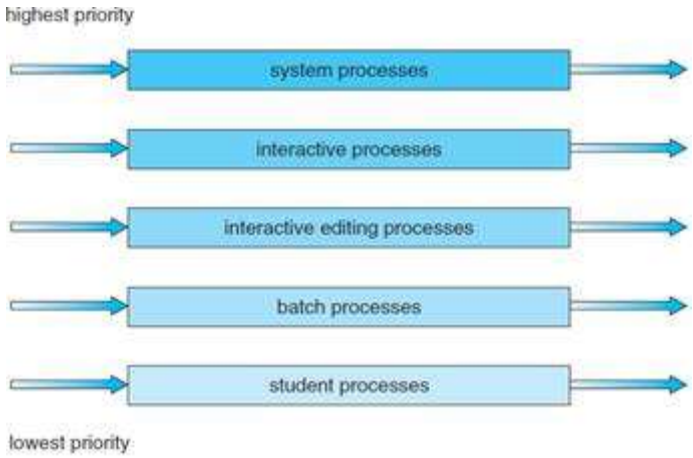
Công thức tính RR (Response Ratio) của thuật toán HRRN.

- Ưu điểm :
    - Không xảy ra starvation.
    - Tự động cân bằng giữa việc ưu tiên một tiến trình có thời gian thực thi nhỏ và một tiến trình đã ở quá lâu trong hệ thống (aging).
- Nhược điểm :
    - Non-Preemptive.

Multilevel Queue Scheduling :

- Mô tả :
    - Cơ chế hoạt động :
        - Hàng đợi ready được chia thành nhiều hàng đợi riêng biệt theo một số tiêu chuẩn như :
            - Đặc điểm và yêu cầu định thời của process.
            - Foreground (interactive) và background process.
        - Process được gán cố định vào một hàng đợi, mỗi hàng đợi sẽ sử dụng một giải thuật riêng.

        - Có 2TH hệ điều hành định thời cho các hàng đợi :
            - Có một độ ưu tiên cố định cho từng hàng đợi (fixed priority scheduling).
                - Hàng đợi có độ ưu tiên cao hơn phải được chạy xong (empty) trước khi hàng đợi có độ ưu tiên thấp hơn được phép chạy.
                - Nếu có 1 tiến trình đi vào hàng đợi có độ ưu tiên cao hơn trong khi hàng đợi có độ ưu tiên thấp hơn đang được thực thi, hàng đợi có độ ưu tiên thấp hơn đó sẽ bị preempt.
            - Time-slice : Mỗi hàng đợi nhận được một khoảng thời gian chiếm CPU và phân phối cho các process trong hàng đợi khoảng thời gian đó.
    - Chế độ quyết định : Non-Preemptive hoặc Preemptive.



Ví dụ về các hàng đợi được phân cấp trong thuật toán MQS.

- Ưu điểm :
    - Áp dụng nhiều giải thuật định thời cho nhiều loại tiến trình có độ ưu tiên khác nhau.
    - Cho phép các CPU-Bound process được ưu tiên hơn trong việc thực thi -> Thời gian hệ thống thực thi tác vụ được cải thiện.
    - Có thể hoạt động trong cả 2 chế độ : Preemptive và Non-Preemptive.
- Nhược điểm :
    - Các hàng đợi đa cấp này cần được giám sát -> Hao phí tài nguyên hệ thống.
    - Process không thể di chuyển từ hàng đợi này sang hàng đợi khác -> Không linh động.

Multilevel Feedback Queue

- Mô tả :
    - Cơ chế hoạt động :
        - (Tương tự Multilevel Feedback Queue).
        - Điểm khác biệt : Cho phép process nhảy từ queue này đến queue khác.
- Ưu điểm :
    - Thích nghi với các tiến trình. VD : Một tiến trình nếu sử dụng quá nhiều CPU time thì sẽ xếp nó vào queue có độ ưu tiên thấp hơn.
    - Aging. VD : Một process đã xuất hiện lâu mà không được thực thi, sẽ được đưa lên 1 queue có độ ưu tiên cao hơn.
    - Thuật toán chung nhất, có thể được thiết kế để phù hợp với các hệ thống khác biệt.

- Nhược điểm :
  - Tốn tài nguyên hệ thống để duy trì các queue -> Có thể không thích hợp đối với các hệ thống nhỏ.
  - Thiết kế phức tạp.

	Mô tả	Ưu điểm	Nhược điểm
FCFS	Cơ chế thực thi : Tiến trình nào yêu cầu CPU trước sẽ được cấp phát trước. Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O. Chế độ quyết định : Non-Preemptive. Hiện thực : Sử dụng hàng đợi FIFO. Tiến trình đi vào được thêm vào cuối hàng đợi. Tiến trình được lựa chọn để xử lý được lấy từ đầu của queue.	Sẽ không bị starvation. Thuật toán này dễ cài đặt. Code đơn giản.	Thời gian chờ trung bình của FCFS thường khá dài (VD : Một process có burst-time rất dài đến trước, khi đó các process có burst-time nhỏ sẽ phải chờ 1 khoảng thời gian rất lâu mới đến lượt thực thi). Lãng phí thời gian do thời gian phần cứng trống khá nhiều (convoy effect). Non-preemptive. Sẽ không hoạt động tốt trong các hệ thống chia sẻ thời gian (time-sharing system) khi các user đều mong muốn được sử dụng CPU trong một khoảng thời gian và không muốn delay quá lâu.
SJF	Cơ chế thực thi: Định thời công việc ngắn nhất trước (Burst-time nhỏ nhất). Khi CPU được tự do, nó sẽ cấp phát cho tiến trình nào yêu cầu ít thời gian nhất để kết thúc (burst-time nhỏ nhất). Burst-time có được từ việc dự đoán, dựa vào các lần chạy trước của tiến trình. Nếu có 2 tiến trình cùng Burst-time, tiến trình nào vào hàng đợi trước sẽ được chạy trước (không xét độ ưu tiên). Chế độ quyết định: Non-Preemptive.	Tối ưu cho thời gian chờ đợi trung bình tối thiểu với một tập tiến trình cho trước.	Cần phải ước lượng thời gian cần CPU tiếp theo của process (Burst time). Có thể xảy ra starvation nếu số lượng process có burst time nhỏ cần được thực thi quá nhiều.
SRTF	Cơ chế thực thi : (Tương tự SJF). Nếu một tiến trình mới được đưa vào danh sách với	Preemptive. Thời gian đáp ứng nhanh cho các tác vụ nhỏ. Tránh việc một tác vụ lớn độc chiếm CPU.	(Các nhược điểm của SJF). Tăng thời gian hoàn thành trung bình.

	chiều dài sử dụng CPU cho lần tiếp theo nhỏ hơn (lưu ý, chỉ nhỏ hơn, nếu burst-time bằng thì không preempt) thời gian còn lại của tiến trình đang xử lý, nó sẽ dừng hoạt động tiến trình hiện hành (preempt). Chế độ quyết định : Preemptive.	Thời gian chờ đợi trung bình thường sẽ nhỏ hơn SJF	
Priority Scheduling	Cơ chế hoạt động : Mỗi tiến trình sẽ được gán 1 độ ưu tiên. CPU sẽ được cấp cho tiến trình có độ ưu tiên cao nhất. Định thời sử dụng độ ưu tiên có thể là : Preemptive : Khi một tiến trình mới xuất hiện có độ ưu tiên cao hơn, nó sẽ preempt tiến trình đang chạy. Non-Preemptive : Tiến trình đang chạy sẽ tiếp tục chạy. Nếu có 2 tiến trình cùng độ ưu tiên, thì tiến trình nào đến trước sẽ được chạy trước.	Các tác vụ quan trọng sẽ được thực thi trước.	Có thể xảy ra starvation : Các 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).

	Burst-time không được áp dụng để so sánh ở đây. Chế độ quyết định : Non-Preemptive hoặc Preemptive.		
Round Robin	Cơ chế hoạt động : Mỗi tiến trình nhận được một đơn vị nhỏ thời gian CPU (time-slice, quantum time), thông thường từ 10-100msec để thực thi. CPU Schedulers sẽ chọn 1 tiến trình từ ready queue và "lên dây cót" một quantum cho tiến trình, sau đó cho tiến trình chạy. Lúc này, sẽ có 2 khả năng có thể xảy ra : Thời gian chạy > Quantum : Khi đó, tiến trình sẽ bị interrupt và CPU Schedulers sẽ chọn tiếp tiến trình tiếp theo. Thời gian chạy < Quantum : Tiến trình tiếp theo sẽ ngay lập tức được thực thi tiếp (không cần chờ hết quantum time của tiến trình trước), và tiến trình tiếp	Thời gian đáp ứng trung bình thường thấp -> Thích hợp cho các hệ thống time-sharing. Không xảy ra tình trạng starvation.	Thời gian chờ đợi trung bình thường khá lớn. Chuyển ngữ cảnh nhiều -> Hao phí cao. Hiệu suất thuật toán phụ thuộc nhiều vào việc chọn quantum time. Không thể sử dụng thuật toán nếu muốn các ứng dụng có độ ưu tiên khác nhau.

	theo đó cũng được gán 1 quantum time. Phụ thuộc nhiều vào quantum time : Quantum time ngắn thì đáp ứng nhanh, tuy nhiên overhead lớn do chuyển ngữ cảnh nhiều. Quantum time phải > thời gian chuyển ngữ cảnh (context switch). Quantum time dài thì đáp ứng chậm, tuy nhiên thông lượng (throughput) sẽ cao. Và khi quantum time quá lớn RR->FCFS (Quantum time lớn -> Không bao giờ bị ngắt -> Ai vào trước làm trước -> FCFS). Khi cả tiến trình vừa thực thi xong và tiến trình mới cũng arrive vào cùng một thời điểm, thì tiến trình mới sẽ vào hàng đợi trước rồi mới đến tiến trình cũ. Các tiến trình đều có độ ưu tiên giống nhau.		

	Chế độ quyết định : Preemptive.		
HRRN	Cơ chế hoạt động : Chọn process tiếp có giá trị RR (Response Ratio) lớn nhất. Các process ngắn được ưu tiên hơn vì service time (hay burst time) nhỏ.	Không xảy ra starvation. Tự động cân bằng giữa việc ưu tiên một tiến trình có thời gian thực thi nhỏ và một tiến trình đã ở quá lâu trong hệ thống (aging).	Non-Preemptive.
Multilevel Queue Scheduling	Cơ chế hoạt động : Hàng đợi ready được chia thành nhiều hàng đợi riêng biệt theo một số tiêu chuẩn như : Đặc điểm và yêu cầu định thời của process. Foreground (interactive) và background process. Process được gán cố định vào một hàng đợi, mỗi hàng đợi sẽ sử dụng một giải thuật riêng. Có 2TH hệ điều hành định thời cho các hàng đợi : Có một độ ưu tiên cố định cho từng hàng đợi (fixed priority scheduling).	Áp dụng nhiều giải thuật định thời cho nhiều loại tiến trình có độ ưu tiên khác nhau. Cho phép các CPU-Bound process được ưu tiên hơn trong việc thực thi -> Thời gian hệ thống thực thi tác vụ được cải thiện. Có thể hoạt động trong cả 2 chế độ : Preemptive và Non-Preemptive.	Các hàng đợi đa cấp này cần được giám sát -> Hao phí tài nguyên hệ thống. Process không thể di chuyển từ hàng đợi này sang hàng đợi khác -> Không linh động.

	Hàng đợi có độ ưu tiên cao hơn phải được chạy xong (empty) trước khi hàng đợi có độ ưu tiên thấp hơn được phép chạy. Nếu có 1 tiến trình đi vào hàng đợi có độ ưu tiên cao hơn trong khi hàng đợi có độ ưu tiên thấp hơn đang được thực thi, hàng đợi có độ ưu tiên thấp hơn đó sẽ bị preempt. Time-slice : Mỗi hàng đợi nhận được một khoảng thời gian chiếm CPU và phân phối cho các process trong hàng đợi khoảng thời gian đó. Chế độ quyết định : Non-Preemptive hoặc Preemptive.		
Multilevel Feedback Queue	Cơ chế hoạt động : (Tương tự Multilevel Feedback Queue). Điểm khác biệt : Cho phép process nhảy từ queue này đến queue khác.	Thích nghi với các tiến trình. VD : Một tiến trình nếu sử dụng quá nhiều CPU time thì sẽ xếp nó vào queue có độ ưu tiên thấp hơn. Aging. VD : Một process đã xuất hiện lâu mà không được thực thi, sẽ được đưa lên 1 queue có độ ưu tiên cao hơn.	Tốn tài nguyên hệ thống để duy trì các queue -> Có thể không thích hợp đối với các hệ thống nhỏ. Thiết kế phức tạp.

		Thuật toán chung nhất, có thể được thiết kế để phù hợp với các hệ thống khác biệt.	

7. Đặc điểm của định thời trên hệ thống có nhiều bộ xử lý? Khi nào cần phải thực hiện cân bằng tải?

Đặc điểm của định thời trên hệ thống có nhiều bộ xử lý, có 2 cách tiếp cận phổ biến:
- Đa xử lý bất đối xứng:
    + Tất cả các thao tác lập lịch, xử lý I/O được thực hiện bởi một bộ xử lý – master server. Các bộ xử lý còn lại chỉ thực thi user code.
    + Ưu điểm: đơn giản, chỉ 1 bộ xử lý truy xuất dữ liệu hệ thống, không cần chia sẻ dữ liệu.
    + Nhược điểm: master server có thể bị nghẽn, làm giảm hiệu năng của hệ thống.
- Đa xử lý đối xứng
    + Mỗi bộ xử lý tự định thời cho chính nó.
    + Có 2 hướng tiếp cận: tất cả tiểu trình cùng nằm trong 1 ready queue hoặc mỗi bộ xử lý tự tổ chức hàng đợi riêng của nó.

Cần cân bằng tải khi có một bộ xử lý có quá nhiều tải trong khi các bộ xử lý khác rỗi, ta cần đảm bảo các bộ xử lý đều được sử dụng hiệu quả.

8. Đặc điểm định thời theo thời gian thực?

Có 2 dạng hệ thống thời gian thực:
- Soft real time systems: các tác vụ quan trọng sẽ được cấp độ ưu tiên lớn nhất, nhưng không đảm bảo bất cứ điều gì khác.
- Hard real time systems: tác vụ phải hoàn thành trong deadline của nó

9. Mô tả các đặc điểm cơ bản của bộ định thời CFS trên Linux?

Nhân Linux từ 2.6.23 sử dụng bộ định thời CFS:

- Định thời theo lớp:

    + Mỗi lớp được gán một độ ưu tiên cụ thể.

    + Bộ định thời chọn tác vụ có độ ưu tiên cao nhất trong lớp có độ ưu tiên cao nhất.

    + Thời gian sử dụng CPU của mỗi tác vụ không dựa trên quantum time cố định mà dựa trên tỷ lệ giờ CPU.

    + Nhân Linux cài đặt sẵn 2 lớp: default và real-time. Các lớp khác có thể được thêm vào.

- Thời gian sử dụng CPU:

  + Được tính dựa trên giá trị nice được gán cho mỗi tác vụ.

  + Giá trị thấp hơn có độ ưu tiên cao hơn.

- CFS xác định tác vụ được thực thi kế tiếp qua virtual run time.

10. Mô tả các đặc điểm cơ bản của định thời trên Windows?

- Định thời theo độ ưu tiên với chế độ trưng dụng.
- Tác vụ có độ ưu tiên cao nhất luôn được chạy tiếp.
- Tiến trình sẽ được thì thi cho đến khi block bởi system call, hết quantum time, bị thay thể bởi 1 tiến trình khác có độ ưu tiên cao hơn.
- Sử dụng 32 độ ưu tiên, được chia thành 2 lớp: variable và real time.
- Mỗi độ ưu tiên có hàng đợi riêng

11. (Bài tập mẫu) Cho các tiến trình với thông tin ở bảng bên dưới. Biết rằng tất cả các tiến trình đều đến ở thời điểm 0 theo thứ tự từ P1 đến P5. Vẽ giản đồ Gantt, tính thời gian đợi trung bình và thời gian lưu lại trong hệ thống (turnaround time) trung bình cho các giải thuật sau:

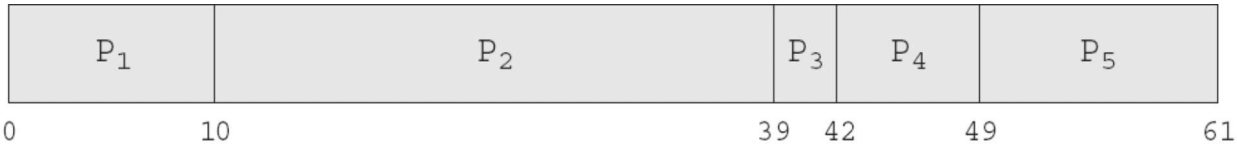
a. FCFS

b. SJF

c. RR với quantum time = 10

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

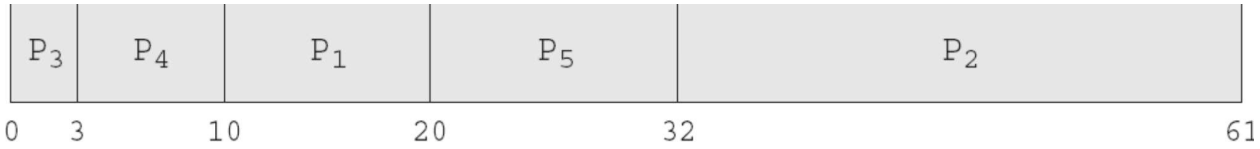
FCFS:



Thời gian đợi trung bình: (0 + 10 + 39 + 42 + 49)/5 = 28

Thời gian lưu lại trong hệ thống trung bình: (10 + 39 + 42 + 49 + 61)/5 = 40.2

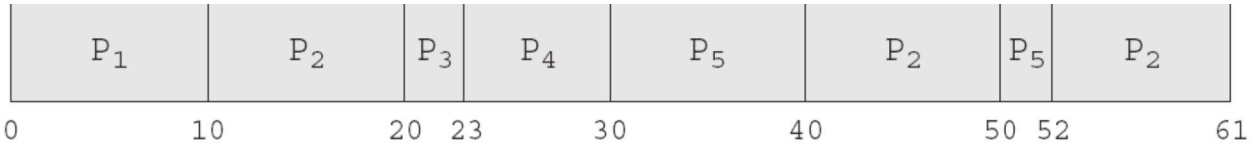
SJF



Thời gian đợi trung bình: (10 + 32 + 0 + 3 + 20)/5 = 13

Thời gian lưu lại trong hệ thống trung bình: (20 + 61 + 3 + 10 + 32)/5 = 25.2

RR với quantum time = 10



Thời gian đợi trung bình: (0 + (10 + 20 + 2) + 20 + 23 + (30 + 10))/5 = 23

Thời gian lưu lại trong hệ thống trung bình: (10 + 61 + 23 + 30 + 52)/5 = 35.2

12. Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào hàng đợi ready và thời gian cần CPU tương ứng như bảng sau:

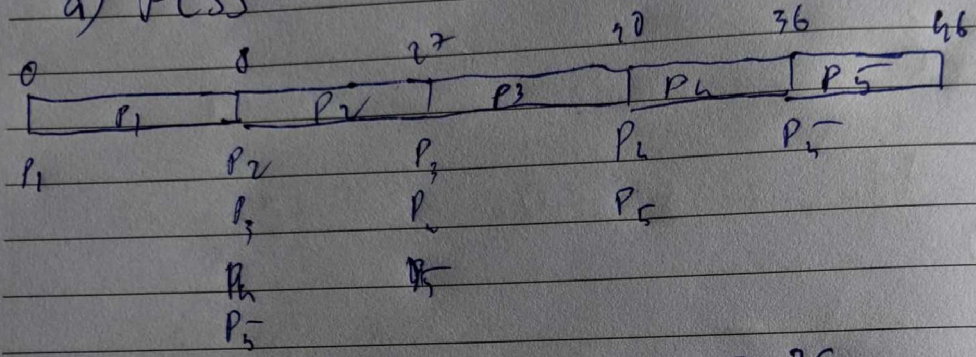
Process	Arrival Time	CPU Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	10

Vẽ sơ đồ Gantt và tính thời gian chờ trung bình, thời gian đáp ứng trung bình, thời gian lưu lại trong hệ thống (turnaround time) trung bình cho các giải thuật sau:

a. FCFS
b. SJF preemptive
c. RR với quantum time = 6.

P_5

a) FCSS



P_1 P_2 P_3 P_4 P_5

P_2 P_3 P_4 P_5

P_3 P_4 P_5

P_4 P_5

P_5

Thời gian đáp ứng: 0, 6, 23, 25, 29

Thời gian đáp ứng trung bình: 16,6

Thời gian chờ: 0, 6, 23, 25, 29

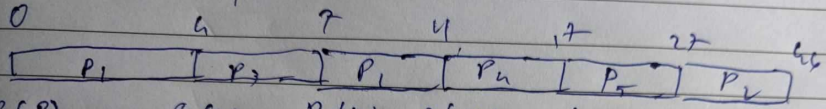
Thời gian chờ trung bình: 16,6

Thời gian hoàn thành: 8, 25, 26, 31, 39

Thời gian hoàn thành trung bình: 25,8

Date . . . No

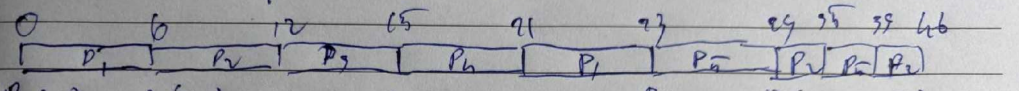
b) STF preemptive



P1(8) P2(9) P2(15) P2(19) P2(14) P2(13)
P3(3) P4(6) P4(6) P5(10)
P1(4) P5(10) P5(10)
P1(4)

Thời gian đáp ứng: 0, 25, 0, 6, 10
Thời gian đáp ứng trung bình: 8, 2
Thời gian chờ: 3, 25, 0, 6, 10
Thời gian chờ trung bình: 8, 8
Thời gian hoàn thành: 11, 44, 3, 12, 20
Thời gian hoàn thành trung bình: 18.

c) RR với quantum time = 6



P1(8) P2(9) P3(3) P4(6) P5(10) P2(3) P5(4) P2(7)
P3(3) P4(6) P1(2) P2(13) P5(4) P2(7)
P4(6) P1(2) P5(10)
P1(2) P5(10) P2(13)
P2(13)

Thời gian đáp ứng: 0, 4, 8, 10, 16
Thời gian đáp ứng trung bình: 7, 6
Thời gian chờ: 15, 25, 8, 10, 22
Thời gian chờ trung bình: 16
Thời gian hoàn thành: 23, 44, 11, 16, 32
Thời gian hoàn thành trung bình: 25, 2

13. (Bài tập mẫu) Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào hàng đợi ready và thời gian cần CPU tương ứng như bảng sau:

Process	Arrival Time	Burst Time
P1	0	13
P2	4	9
P3	6	4
P4	7	20
P5	12	10

Vẽ giản đồ Gantt và tính thời gian đợi trung bình, thời gian đáp ứng trung bình, thời gian lưu lại trong hệ thống (turnaround time - thời gian hoàn thành) trung bình khi thực hiện các giải thuật định thời sau:

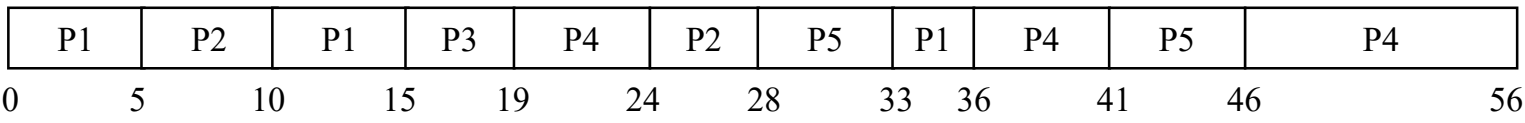
a) Round Robin với quantum time = 5

b) SRTF

Có nhận xét gì về tính hiệu quả của hai giải thuật trên?

a. Round Robin với quantum time = 5

Giản đồ Gantt:



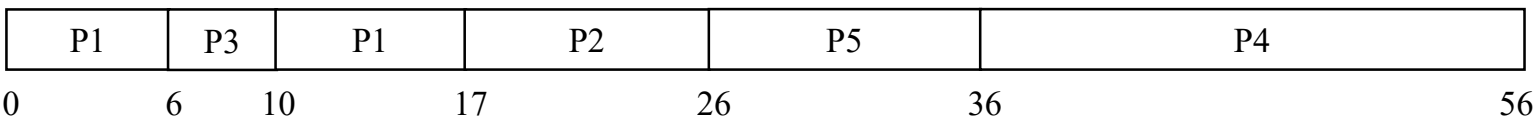
Thời gian đáp ứng trung bình: (0 + 1 + 9 + 12 + 16)/5 = 7.6

Thời gian đợi trung bình: ((5 + 18) + (1 + 14) + 9 + (12 + 12 + 5) + (16 + 8))/5 = 20

Thời gian hoàn thành trung bình: (36 + 24 + 13 + 49 + 34)/5 = 31.2

b. SRTF

Giản đồ Gantt:



Thời gian đáp ứng trung bình: (0 + 13 + 0 + 29 + 14)/5 = 11.2

Thời gian đợi trung bình: (4 + 13 + 0 + 29 + 14)/5 = 12

Thời gian hoàn thành trung bình: (17 + 22 + 4 + 49 + 24)/5 = 23.2

Nhận xét về hai giải thuật trên:

- SRTF hiệu quả hơn (tốt hơn) Round Robin nếu xét trên các tiêu chuẩn thời gian đợi (trung bình) và thời gian hoàn thành (trung bình).

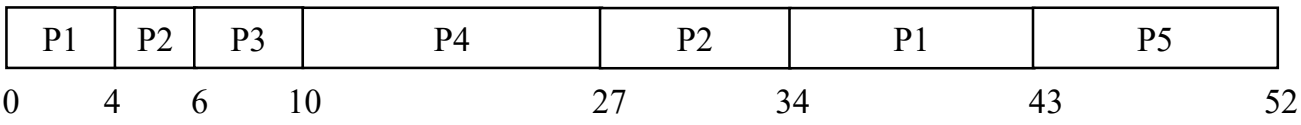
- Round Robin cho thời gian đáp ứng (trung bình) tốt hơn SRTF.

14. (Bài tập mẫu) Cho 5 tiến trình P1, P2, P3, P4, P5 với thời gian vào hàng đợi ready và thời gian cần CPU tương ứng như bảng sau:

Process	Arrival Time	Burst Time	Priority
P1	0	13	4
P2	4	9	3
P3	6	4	1
P4	7	17	2
P5	12	9	5

Vẽ giản đồ Gantt và tính thời gian đợi trung bình, thời gian đáp ứng trung bình, thời gian lưu lại trong hệ thống (turnaround time - thời gian hoàn thành) trung bình khi thực hiện giải thuật định thời Preemptive Priority (độ ưu tiên 1 > 2 > 3 …)

Giản đồ Gantt:



Thời gian đáp ứng trung bình: (0 + 0 + 0 + 3 + 31)/5 = 6.8

Thời gian đợi trung bình: (30 + 21 + 0 + 3 + 31)/5 = 17

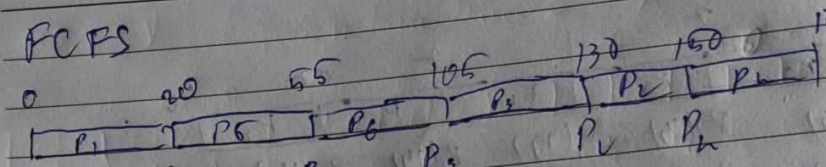
Thời gian hoàn thành trung bình: (43 + 30 + 4 + 20 + 40)/5 = 27.4

15. Sử dụng các giải thuật FCFS, SJF, SRTF, Priority -Pre, RR (10) để tính các giá trị thời gian đợi, thời gian đáp ứng, thời gian hoàn thành trung bình và vẽ giản đồ Gantt cho các tiến trình sau:

Process	Arrival Time	Burst Time	Priority
P1	0	20	20
P2	25	25	30
P3	20	25	15
P4	35	15	35
P5	10	35	5
P6	15	50	10

Date . . . No

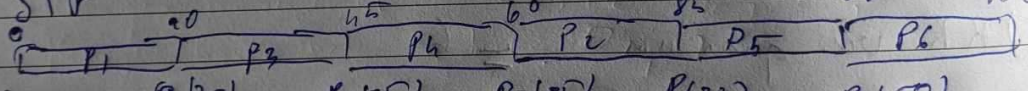
FCFS



P1 P5 P6 P3 P2 P4
P6 P3 P2 P4
P3 P2 P4
P4

Thời gian đáp ứng: 0, 105, 85, 120, 10, 40
Thời gian đáp ứng trung bình: 60
Thời gian chờ: 0, 105, 85, 120, 10, 40
Thời gian chờ trung bình: 60
Thời gian hoàn thành: 20, 130, 110, 135, 45, 90
Thời gian hoàn thành trung bình: 88,3

STF

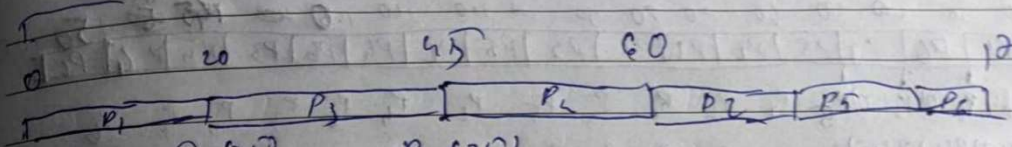


P1(20) P3(35) P4(35) P2(35) P5(33) P6(60)
P6(60) P6(60) P6(50) P6(50)
P3(25) P2(25) P4(25) B
P4(15)

Thời gian đáp ứng: 0, 35, 0, 10, 75, 105
Thời gian đáp ứng trung bình: 37,5
Thời gian chờ: 0, 35, 0, 10, 75, 105
Thời gian chờ trung bình: 37,5
Thời gian hoàn thành: 20, 60, 25, 25, 110, 155
Thời gian hoàn thành trung bình: 65,8

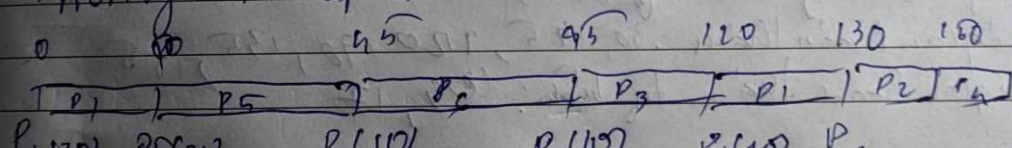
Date . . . No .

SRTF



$P_1(20)$ $P_5(35)$ $P_5(35)$
$P_6(50)$ $P_6(50)$
$P_3(25)$ $P_2(25)$
$P_4(15)$

Priority - Preemptive

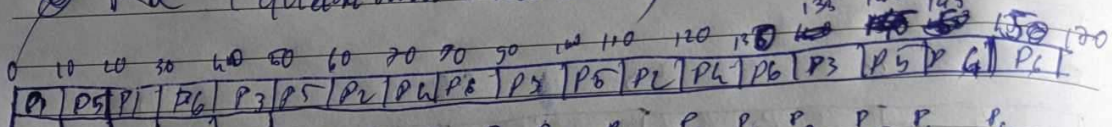


$P_1(20)$ $P_5(35)$ $P_2(10)$ $P_6(10)$ $P_1(10)$ P_2
$P_2(10)$ $P_6(50)$ $P_3(25)$ $P_2(25)$ P_4
$P_3(25)$ $P_2(25)$ $P_4(15)$
$P_2(25)$ $P_4(15)$
$P_4(15)$

Thời gian đợi trung bình: 0, 105, 75, 120, 0, 30
Thời gian đợi trung bình: 55
Thời gian chờ: 100, 105, 75, 120, 0, 30
Thời gian chờ trung bình: 73,3
Thời gian hoàn thành: 130, 130, 100, 135, 35, 50
Thời gian hoàn thành trung bình: 101,67

Date . . . No

RR (quantum time = 10)



Thời gian chờ: 0, 7.5, 20, 25, 0, 15

Thời gian chờ trung bình: 12,5

Thời gian chờ: 10, 140, 45, 75, 100, 105

Thời gian chờ trung bình: 80, 83

Thời gian hoàn thành: 30, 125, 220, 190, 135, 155

Thời gian hoàn thành trung bình: 165, 17

16. Xét tập các tiến trình sau (với thời gian yêu cầu CPU và độ ưu tiên kèm theo). Vẽ giản đồ Gantt và tính thời gian đợi trung bình và thời gian lưu lại trong hệ thống trung bình (turnaround time) cho các giải thuật sau:
    a. SJF Preemptive
    b. RR với quantum time = 2
    c. Preemptive Priority (độ ưu tiên 1 > 2 > ...)

Process	Arrival Time	Burst Time	Priority
P1	0	10	3
P2	1	3	2
P3	2	2	1
P4	3	1	2
P5	4	5	4

17. Cho 5 tiến trình với thời gian vào hàng đợi ready và thời gian cần CPU tương ứng như bảng sau:

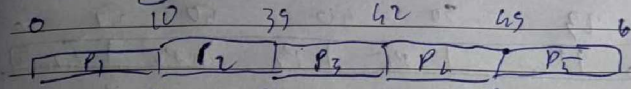
Process	Arrival Time	Burst Time
P1	0	10
P2	2	29
P3	4	3
P4	5	7
P5	7	12

Vẽ giản đồ Gantt và tính thời gian đợi trung bình, thời gian đáp ứng trung bình và thời gian lưu lại trong hệ thống (turnaround time) trung bình cho các giải thuật sau:

a. FCFS
b. SJF preemptive
c. RR với quantum time = 10

Date . . . No

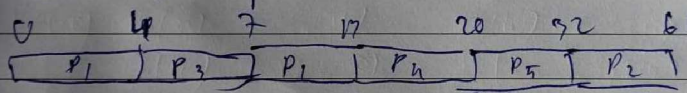
FCFS:



P1 P2 P3 P4 P5
P2 P3 P4 P5
P3 P4 P5
P4 P5
P5

Thời gian đáp ứng: 0, 8, 35, 37, 42
Thời gian đáp ứng trung bình: 24,4
Thời gian chờ : 0, 8, 35, 37, 42
Thời gian chờ trung bình: 24,4
Thời gian hoàn thành: 10, 37, 38, 44, 54
Thời gian hoàn thành trung bình: 36,6

SJF Preemptive

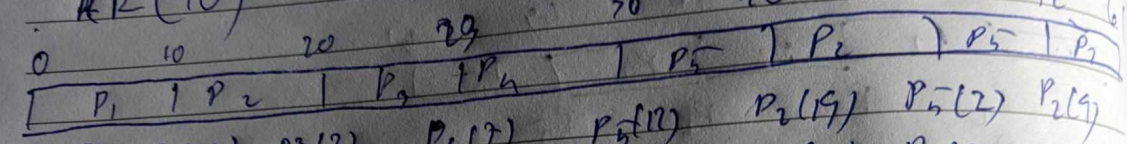


P1(10) P2(29) P2(29) P2(23) P2(29) P2(29)
P3(8) P4(6) P4(7) P5(12)
P1(6) P4(7) P5(12)
P5(12)

Thời gian đáp ứng: 0, 30, 0, 8, 13
Thời gian đáp ứng trung bình: 10,2
Thời gian chờ: 3, 30, 0, 8, 13
Thời gian chờ trung bình: 10,8
Thời gian hoàn thành: 13, 59, 3, 15, 25
Thời gian hoàn thành trung bình: 23

Date . . . No

RR (10)



$P_1(10)$ $P_2(29)$ $P_3(3)$ $P_4(7)$ $P_5(12)$ $P_2(19)$ $P_5(2)$ $P_2(9)$
$P_3(7)$ $P_4(7)$ $P_5(12)$ $P_2(19)$ $P_5(2)$ $P_2(9)$
$P_4(7)$ $P_5(12)$ $P_2(9)$
$P_5(12)$ $P_2(19)$

Thời gian đáp ứng: 0, 8, 16, 18, 23
Thời gian đáp ứng trung bình: 13
Thời gian chờ: 0, 30, 16, 18, 31
Thời gian chờ trung bình: 19, 4
Thời gian hoàn thành: 10, 59, 19, 25, 45
Thời gian hoàn thành trung bình: 31, 6