

Tên: Lê Anh Thư
MSSV: 20521985

MÔN HỌC: HỆ ĐIỀU HÀNH

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4

Câu 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).
 - Lờ 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.

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

Định thời CPU là

Chọn một process (từ ready queue) thực thi

Với một multithreaded kernel, việc định thời CPU là do OS chọn kernel thread được chiếm CPU

Bộ định thời chịu trách nhiệm thực hiện việc này: Short-term scheduling (hay còn gọi là Dispatcher)

Câu 3: Phí tổn gây ra khi định thời là gì?

- Bộ định thời sẽ chuyển quyền điều khiển CPU về cho process được chọn.
- Bao gồm:
 - Chuyển ngữ cảnh (sử dụng thông tin ngữ cảnh trong PCB)
 - Chuyển chế độ người dùng
 - 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 bộ định thời dừng một process và khởi động một process khác

Câu 4: Trình bày các tiêu chuẩn định thời CPU?

- Hướng người dùng (User-oriented)

Thời gian đáp ứng (Response time): khoảng thời gian process nhận yêu cầu đến khi yêu cầu đầu tiên được đáp ứng (time-sharing, interactive system) → cực tiểu

Thời gian quay vòng (hoàn thành) (Turnaround time): khoảng thời gian từ lúc một process được nạp vào hệ thống đến khi process đó kết thúc → cực tiểu

Thời gian chờ (Waiting time): tổng thời gian một process đợi trong ready queue → cực tiểu

- Hướng hệ thống (System-oriented)

Sử dụng CPU (processor utilization): định thời sao cho CPU càng bận càng tốt → cực đại

Công bằng (fairness): tất cả process phải được đối xử như nhau

Thông lượng (throughput): số process hoàn tất công việc trong một đơn vị thời gian → cực đại

Câu 5: Kể tên các giải thuật định thời CPU?

First-Come, First-Served (FCFS).

Shortest-Job-First Scheduling (SJF).

Preemptive SJF (hay Shortest-Remaining-Time First – SRTF).

Priority Scheduling.

Round-Robin (RR).

Highest Response Ratio Next (HRRN).

Multilevel Queue.

Multilevel Feedback Queue.

Câu 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.

Giải thuật	Mô tả	Ưu điểm	Nhược điểm
1. First-Come, First-	Cơ chế thực thi : <ul style="list-style-type: none">- Tiến trình nào yêu cầu CPU trước sẽ được cấp phát trước.	Sẽ không bị starvation	Thời gian chờ trung bình của FCFS thường khá dài (VD : Một process có

<p>Served (FCFS)</p>	<ul style="list-style-type: none"> - Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O. <p>Chế độ quyết định : Non-Preemptive.</p> <p>Hiện thực : Sử dụng hàng đợi FIFO.</p> <ul style="list-style-type: none"> - Tiến trình đi vào được thêm vào cuối hàng đợi. - Tiến trình được lựa chọn để xử lý được lấy từ đầu của queue. 	<p>Thuật toán này dễ cài đặt. Code đơn giản.</p>	<p>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).</p> <p>Lãng phí thời gian do thời gian phần cứng trống khá nhiều (convoy effect).</p> <p>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.</p>
<p>2. Shortest Job First (SJF)</p>	<p>Cơ chế thực thi :</p> <ul style="list-style-type: none"> - 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 	<p>Tối ưu.</p> <p>Cho thời gian chờ đợi trung bình tối thiểu với một tập tiến trình cho trước.</p>	<p>Cần phải ước lượng thời gian cần CPU tiếp theo của process (Burst time).</p> <p>Có thể xảy ra starvation nếu số lượng process có burst time nhỏ cần được thực thi quá nhiều.</p>

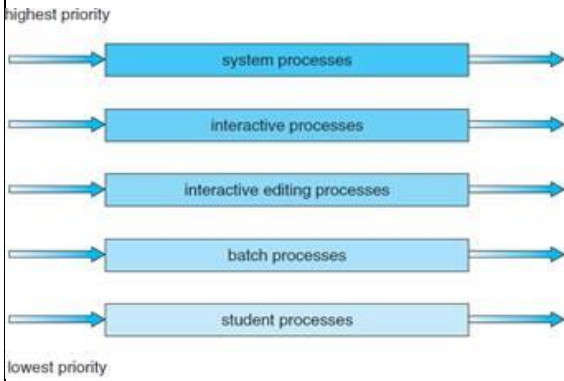
	<p>hàng đợi trước sẽ được chạy trước (không xét độ ưu tiên).</p> <p>Chế độ quyết định : Non-Preemptive.</p>		
<p>3. Shortest Remaining Time First (SRTF)</p>	<p>Cơ chế thực thi :</p> <ul style="list-style-type: none"> - Tương tự SJF). - Nếu một tiến trình mới được đưa vào danh sách với chiều dài sử dụng 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). <p>Chế độ quyết định : Preemptive.</p>	<p>Preemptive . Thời gian đáp ứng nhanh cho các tác vụ nhỏ.</p> <p>Tránh việc một tác vụ lớn độc chiếm CPU.</p> <p>Thời gian chờ đợi trung bình thường sẽ nhỏ hơn SJF.</p>	<p>(Các nhược điểm của SJF).</p> <p>Tăng thời gian hoàn thành trung bình.</p>
<p>4. Priority Scheduling</p>	<p>Cơ chế hoạt động :</p> <ul style="list-style-type: none"> - 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 	<p>Các tác vụ quan trọng sẽ được thực thi trước.</p>	<p>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).</p>

	<p>hơn, nó sẽ preempt tiến trình đang chạy.</p> <p>Non-Preemptive : Tiến trình đang chạy sẽ tiếp tục chạy.</p> <ul style="list-style-type: none"> - 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. <p>Chế độ quyết định : Non-Preemptive hoặc Preemptive.</p>		
5. Round-Robin (RR)	<p>Cơ chế hoạt động :</p> <ul style="list-style-type: none"> - 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 : <ul style="list-style-type: none"> 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 	<p>Thời gian đáp ứng trung bình thường thấp -> Thích hợp cho các hệ thống time-sharing.</p> <p>Không xảy ra tình trạng starvation.</p>	<p>Thời gian chờ đợi trung bình thường khá lớn.</p> <p>Chuyển ngữ cảnh nhiều -> Hao phí cao.</p> <p>Hiệu suất thuật toán phụ thuộc nhiều vào việc chọn quantum time.</p> <p>Không thể sử dụng thuật toán nếu muốn các ứng dụng có độ ưu tiên khác nhau.</p>

	<p>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.</p> <ul style="list-style-type: none"> - Phụ thuộc nhiều vào quantum time : <p>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).</p> <p>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).</p> - Khi cả tiến trình vừa thực thi xong và tiến trình mới cũng arrive vào cùng một thời điểm, thì tiến trình mới sẽ 		
--	---	--	--

	<p>vào hàng đợi trước rồi mới đến tiến trình cũ.</p> <ul style="list-style-type: none"> - Các tiến trình đều có độ ưu tiên giống nhau. <p>Chế độ quyết định : Preemptive.</p>		
<p>6. Highest Response Ratio Next (HRRN)</p>	<p>Cơ chế hoạt động :</p> <ul style="list-style-type: none"> - 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. <p>Công thức :</p> $response\ ratio = \frac{waiting\ time + estimated\ run\ time}{estimated\ run\ time} = 1 + \frac{waiting\ time}{estimated\ run\ time}$	<p>Không xảy ra starvation.</p> <p>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).</p>	<p>Phải biết trước burst time của process.</p> <p>Non-Preemptive.</p>
<p>7. Multilevel Queue</p>	<p>Cơ chế hoạt động :</p> <ul style="list-style-type: none"> • 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. 	<p>Áp dụng nhiều giải thuật định thời cho nhiều loại tiến trình có độ ưu tiên khác nhau.</p>	<p>Các hàng đợi đa cấp này cần được giám sát -> Hao phí tài nguyên hệ thống.</p> <p>Process không thể di chuyển từ hàng đợi này sang hàng đợi khác -> Không linh động.</p>

	<ul style="list-style-type: none"> • Process được gán cố định vào một hàng đợi, mỗi hàng đợi sẽ sử dụng một giải thuật riêng. <p>Có 2TH hệ điều hành định thời cho các hàng đợi :</p> <ul style="list-style-type: none"> - 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. <p>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.</p> <p>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.</p> <p>Chế độ quyết định : Non-Preemptive hoặc Preemptive.</p>	<p>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.</p> <p>Có thể hoạt động trong cả 2 chế độ :</p> <p>Preemptive và Non-Preemptive</p> <p>.</p>	
--	--	--	--

	 <p>highest priority</p> <p>lowest priority</p> <p>Ví dụ về các hàng đợi được phân cấp trong thuật toán MQS.</p>		
8. Multilevel Feedback Queue	<p>Cơ chế hoạt động :</p> <ul style="list-style-type: none"> - (Tương tự Multilevel Feedback Queue). - Điểm khác biệt : Cho phép process nhảy từ queue này đến queue khác. 	<p>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.</p> <p>Aging. VD : Một process đã xuất hiện lâu mà không được thực thi, sẽ được đưa lên 1 queue</p>	<p>Tốn tài nguyên hệ thống để duy trì các queue -> Có thể không thích hợp đối với các hệ thống nhỏ.</p> <p>Thiết kế phức tạp.</p>

		<p>có độ ưu tiên cao hơn.</p> <p>Thuật toán chung nhất, có thể được thiết kế để phù hợp với các hệ thống khác biệt.</p>	
--	--	---	--

Câu 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:
Định thời CPU trở nên phức tạp hơn khi hệ thống có nhiều bộ xử lý.

Khái niệm đa bộ xử lý có thể là một trong các dạng sau:

- CPU có nhiều lõi vật lý (Multicore CPUs)
- CPU có nhiều luồng xử lý trên một lõi (Multithreaded cores)
- Hệ thống NUMA (non-uniform memory access)
- Đa xử lý không đồng nhất (Heterogeneous multiprocessing)

Có hai cách tiếp cận phổ biến: đa xử lý bất đối xứng (asymmetric multiprocessing) và đa xử lý đối xứng (symmetric multiprocessing - SMP).

- Cần phải thực hiện cân bằng tải khi:

Một bộ xử lý có quá nhiều tải, trong khi các bộ xử lý khác rỗi. Cần phải đảm bảo các bộ xử lý đều được sử dụng hiệu quả.

Mục tiêu của cân bằng tải là phân phối khối lượng công việc (workload) đều nhau cho các CPU.

Câu 8: Đặc điểm định thời theo thời gian thực?

Đặc điểm định thời theo thời gian thực:

Có nhiều thách thức do yêu cầu về tính chất 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

Câu 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 (Completely Fair Scheduler)

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, có giá trị từ -20 đến 19.

Giá trị thấp hơn có độ ưu tiên cao hơn.

Target latency – khoảng thời gian mà một tiến trình cần được chạy ít nhất một lần.

Target latency có thể tăng lên nếu số lượng tiến trình tăng lên.

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

Mỗi tác vụ có giá trị virtual run time riêng, được kết hợp với một hệ số đặc biệt dựa trên độ ưu tiên.

Các tiến trình có độ ưu tiên bình thường có virtual run time tương đương với thời gian chạy thực tế.

Chọn tiến trình có virtual run time nhỏ nhất để thực thi tiếp.

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

- Định thời theo độ ưu tiên với chế độ trung dụng.
- Tác vụ có độ ưu tiên cao nhất luôn được chạy tiếp.
- Tiến trình sẽ được thực thi cho đến khi (1) block bởi system call, (2) hết quantum time, (3) bị thay thế bởi một tiến trình khác có độ ưu tiên cao hơn.
- Sử dụng 32 độ ưu tiên, được chia thành 2 lớp: variable (1-15) và real-time (16-31). Độ ưu tiên 0 dành cho quản lý bộ nhớ.
- Mỗi độ ưu tiên có hàng đợi riêng.
- Idle thread được chạy nếu không có bất cứ tác vụ nào trong hàng đợi.
- Các hàm thư viện Windows API cung cấp cho tiến trình các lớp ưu tiên sau:
 - o REALTIME_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, IDLE_PRIORITY_CLASS.
- Tiến trình có thể có các độ ưu tiên tương đối sau:
 - o TIME_CRITICAL, HIGHEST, ABOVE_NORMAL, NORMAL, BELOW_NORMAL, LOWEST, IDLE
- Lớp ưu tiên và độ ưu tiên tương đối có thể kết hợp để xác định giá trị ưu tiên.
- Độ ưu tiên cơ sở (lúc khởi tạo) là NORMAL bên trong lớp.
- Khi hết quantum, độ ưu tiên có thể giảm nhưng không nhỏ hơn độ ưu tiên cơ sở.
- Các độ ưu tiên trên Windows

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

- Windows 7 có thêm user-mode scheduling (UMS):

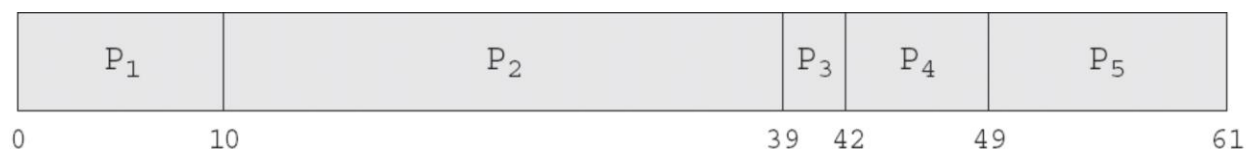
- Ứng dụng tạo và quản lý tiểu trình độc lập với nhân.
- Hiệu quả hơn trong trường hợp có nhiều tiểu trình.
- Định thời UMS được thực hiện với sự hỗ trợ của các thư viện như C++ Concurrent Runtime (ConcRT).

Câu 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:

- FCFS
- SJF
- RR với quantum time = 10

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

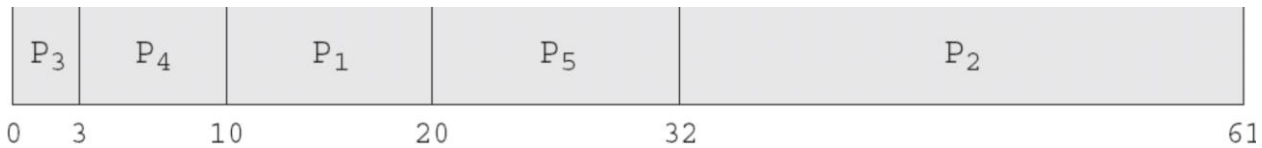
FCFS:



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

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

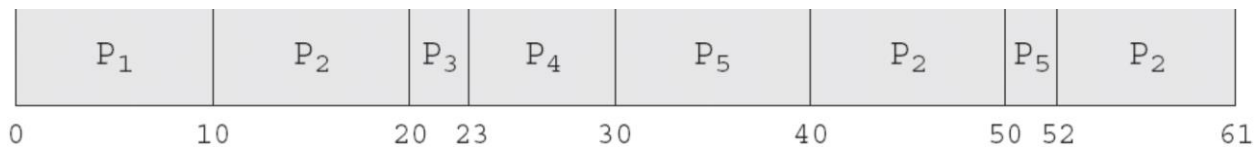
SJF



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: $(10 + 61 + 23 + 30 + 52)/5 = 35.2$

Câu 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:

- FCFS
- SJF preemptive
- RR với quantum time = 6.

- **FCFS:**

Sơ đồ Gantt

P1	P2	P3	P4	P5	
0	8	27	30	36	46

Thời gian chờ:

P1: 0

P2: 6

P3: 23

P4: 25

P5: 29

Thời gian chờ trung bình: $(0 + 6 + 23 + 25 + 29)/5 = 16,6$

Thời gian đáp ứng:

P1: 0

P2: 6

P3: 23

P4: 25

P5: 29

Thời gian đáp ứng trung bình: $(0 + 6 + 23 + 25 + 29)/5 = 16,6$

Thời gian hoàn thành:

P1: 8

P2: 25

P3: 26

P4: 31

P5: 39

Thời gian hoàn thành trung bình: $(8 + 25 + 26 + 31 + 39)/5 = 25,8$

- **SJF preemptive:**

Gantt

P1	P3	P1	P4	P5	P2	
0	4	7	11	17	27	46

Thời gian chờ:

P1: $0 + 3 = 3$

P2: 25

P3: 0

P4: 6

P5: 10

Thời gian chờ trung bình: $(3 + 25 + 0 + 6 + 10)/5 = 8,8$

Thời gian đáp ứng:

P1: 0

P2: 25

P3: 0

P4: 6

P5: 10

Thời gian đáp ứng trung bình: $(0 + 25 + 0 + 6 + 10)/5 = 8,2$

Thời gian hoàn thành:

P1: 11

P2: 44

P3: 3

P4: 12

P5: 20

Thời gian hoàn thành trung bình: $(11 + 44 + 3 + 12 + 20)/5 = 18$

$\nearrow 7$
 Round Robin: $P1 - 7$ $P2 - 13$ $P5 - 4$

Gantt

	P1	P2	P3	P4	P5	P1	P2	P5	P2
0	6	12	15	21	27	29	35	39	46

Thời gian chờ:

P1: $0 + 21 = 21$

P2: $4 + 17 + 4 = 25$

P3: 8

P4: 10

P5: $14 + 8 = 22$

Thời gian chờ trung bình: $(21 + 25 + 8 + 10 + 22)/5 = 17,2$

Thời gian đáp ứng:

P1: 0

P2: 4

P3: 8

P4: 10

P5: 14

Thời gian đáp ứng trung bình: $(0 + 4 + 8 + 10 + 14)/5 = 7,2$

Thời gian hoàn thành:

P1: 29

P2: 44

P3: 11

P4: 16

P5: 22

Thời gian chờ trung bình: $(29 + 44 + 11 + 16 + 22)/5 = 24,4$

Câu 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:

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

0 5 10 15 19 24 28 33 36 41 46
56

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:

P1	P3	P1	P2	P5	P4	
0	6	10	17	26	36	56

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:

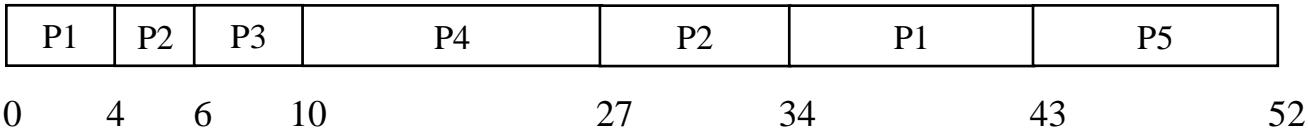
- SRTF hiệu quả hơn (tốt hơn) Round Robin nếu xét trên các tiêu chuẩn thời gian đợi (trung bình) và thời gian hoàn thành (trung bình).
- Round Robin cho thời gian đáp ứng (trung bình) tốt hơn SRTF.

Câu 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 $1 > 2 > 3 \dots$)

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: $(43 + 30 + 4 + 20 + 40)/5 = 27.4$

Câu 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

BA15

FCFS

P1	P5	P6	P3	P2	P4
0	20	55	105	130	155

- Thời gian chờ trung bình: $(0 + 10 + 40 + 85 + 105 + 120) / 6 = 60$

- Thời gian đáp ứng trung bình: $(0 + 10 + 40 + 85 + 105 + 120) / 6 = 60$

- Thời gian hoàn thành trung bình: $(20 + 25 + 90 + 110 + 120 + 135) / 6 = 88,3$

SJF

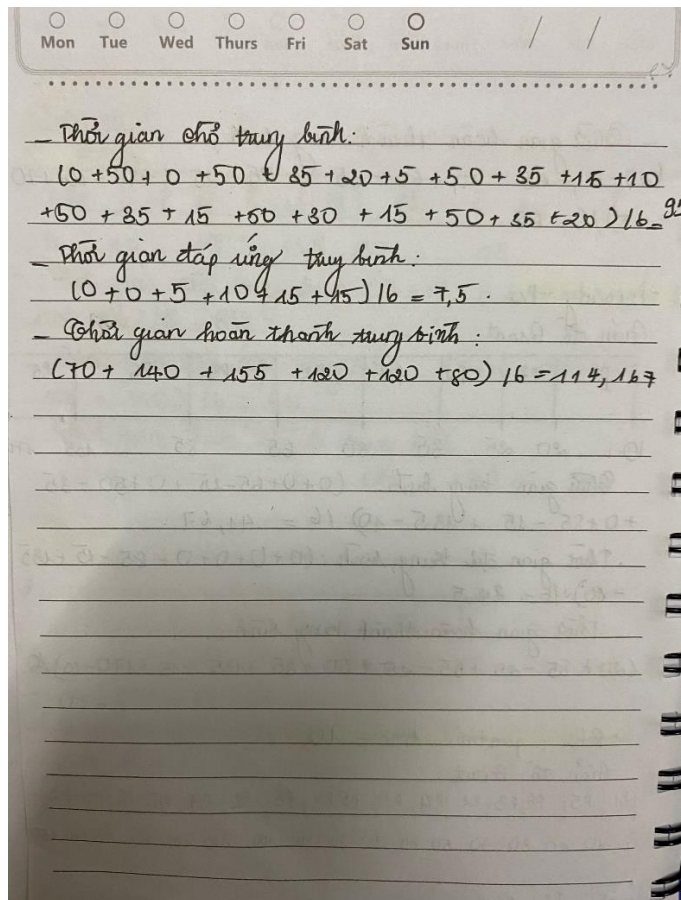
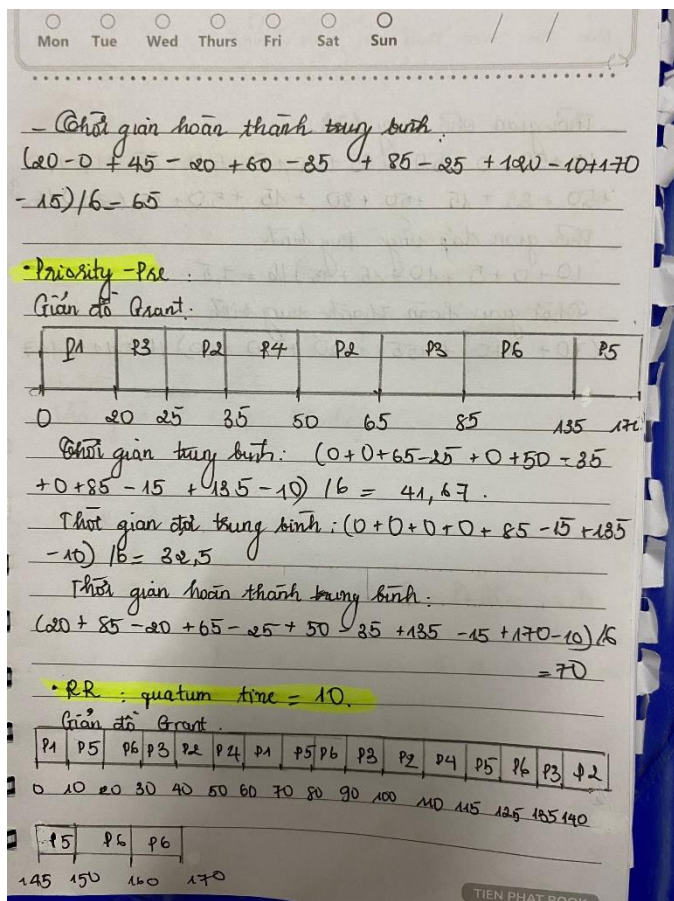
P1	P3	P4	P2	P5	P6
0	20	45	60	85	120

- Thời gian chờ trung bình:

$$(0 + 20 + 20 + 45 + 35 + 60 - 25 + 85 - 10 + 120 - 15) / 6 = 37,5$$

- Thời gian đáp ứng trung bình:

$$(0 + 20 - 20 + 45 - 35 + 60 - 25 + 85 - 10 + 120 - 15) / 6 = 37,5$$



Câu 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:

- SJF Preemptive
- RR với quantum time = 2
- Preemptive Priority (độ ưu tiên $1 > 2 > \dots$)

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
----	---	---	---

Mon Tue Wed Thurs Fri Sat Sun

Bài 16:

a) SJF Preemptive

P1, P2, P4, P3, P5, P1

0 1 4 5 7 12 12 21

Thời gian chờ trung bình: $(0 + 11 + 0 + 3 + 0 + 1 + 3) / 5 = 3,6$

Thời gian luân lại trung bình:

b) RR với quantum time = 2

Gián đồ Gantt:

P1 P2 P3 P4 P5 P1 P2 P5 P1 P5 P1

0 2 4 6 7 9 11 12 14 16 17 21

Thời gian chờ trung bình: $(0 + 7 + 3 + 1 + 1 + 7 + 2 + 3 + 2 + 3 + 3 + 2) / 15 = 6,4$

Mon Tue Wed Thurs Fri Sat Sun

b) RR: quantum time = 2

P1 P2 P3 P1 P4 P5 P2 P1 P5 P1

0 2 4 6 8 9 11 12 14 16 18

P5, P1

19 21

Thời gian chờ tb: $(11 + 8 + 2 + 5 + 10) / 5 = 7,2$

Thời gian hoàn thành trung bình: $(21 + 11 + 4 + 6 + 15) / 5 = 11,4$

c) Preemptive Priority

P1 P2 P3 P2 P4 P1 P5

0 1 2 4 6 7 16 21

Thời gian chờ trung bình: $(6 + 2 + 0 + 3 + 12) / 5 = 7,6$

Thời gian hoàn thành trung bình:

$(16 + 5 + 2 + 4 + 17) / 5 = 8,8$

Câu 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:

- FCFS
- SJF preemptive
- RR với quantum time = 10

Mon Tue Wed Thurs Fri Sat Sun / /

Bài 17:

a) FCFS

P1	P2	P3	P4	P5
0	10	39	42	49
				61

- Thời gian đáp ứng:

$P_1 = 0, P_2 = 8, P_3 = 35, P_4 = 37, P_5 = 42$

- Thời gian đáp ứng trung bình: $(0 + 8 + 35 + 37 + 42) / 5 = 24,4$

- Thời gian chờ:

$P_1 = 0, P_2 = 8, P_3 = 35, P_4 = 37, P_5 = 42$

- Thời gian chờ TB $(0 + 8 + 35 + 37 + 42) / 5 = 24,4$

- Thời gian hoàn thành:

$P_1 = 10, P_2 = 37, P_3 = 38, P_4 = 44, P_5 = 54$

- Thời gian hoàn thành trung bình: $(10 + 37 + 38 + 44 + 54) / 5 = 36,6$

b) SJF preemptive

P1	P3	P1	P4	P5	P2
0	4	7	13	20	32
					61

- Thời gian chờ trung bình: $(0 + 3 + 0 + 8 + 13 + 30) / 5 = 10,8$

Mon Tue Wed Thurs Fri Sat Sun / /

- Thời gian đáp ứng trung bình: $(0 + 0 + 8 + 18 + 30) / 5 = 8,2$

- Thời gian hoàn thành trung bình: $(15 + 3 + 15 + 25 + 59) / 5 = 23,6$

c) RR: với quantum time = 10

P1	P2	P3	P4	P5	P2	P5	P2
0	10	20	23	30	40	50	52
							61

- Thời gian chờ trung bình: $(0 + 8 + 20 + 2 + 16 + 18 + 23 + 10) / 5 = 19,4$

- Thời gian đáp ứng trung bình: $(0 + 8 + 16 + 18 + 23) / 5 = 13$

- Thời gian hoàn thành trung bình: $(10 + 39 + 19 + 15 + 25) / 5 = 25,6$