**­­ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

Text

Description automatically generated with medium confidence

**Nguyễn Lê Quỳnh Hương – 21520255**

IT007.N21.HTCL

**BÀI TẬP:**

**CHƯƠNG 4**

**Định thời CPU**

**Giảng viên : Lê Hoài Nghĩa**

**Hồ Chí Minh, 2023**

**Câu 1 : Tại sao phải định thời? Có những loại bộ định thời nào?**

* Trong các hệ thống multiprogramming / 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 giải quyết vấn đề phân loại và lựa chọn process thực thi sao cho hiệu quả nhất, vì vậy cần có chiến lược định thời.

- Các bộ định thời:

* Long-term scheduling: Xác định process mới (new) nào được tiếp tục vào “sâu hơn” trong hệ thống.
* Medium-term scheduling: Xác định process nào được đưa vào (swap in), đưa ra khỏi (swap out) bộ nhớ chính.
* Short-term scheduling: 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.

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

- Định thời CPU:

* 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 là short-term scheduling: xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp (còn được gọi là định thời CPU, CPU scheduling)

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

* Dispatch latency là 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 thành được 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 (SJF)
* Shortest Remaining Time First (SRTF)
* Round-Robin (RR)
* Priority Scheduling
* 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.**

* **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:
* (Tương tự SJF).
* Tăng thời gian hoàn thành trung bình.
* **RR:**
* **Mô tả:**
* Cơ chế thực thi:
* Mỗi tiến trình nhận được một đơn vị nhỏ thời gian CPU (time-slice, quantum time), thông thường từ 10 - 100ms 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 lơn hơn 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.
* Nếu có n process trong hàng đợi ready và quantum time = q thì không có process nào phải chờ đợi quá (n -1)q đơn vị thời gian.
* Chế độ quyết định: Preemptive
* **Ưu điểm:**
* Các tiến trình đều có độ ưu tiên giống nhau.
* 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.
* **Priority Scheduling:**
* **Mô tả:**
* Cơ chế thực thi:
* 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).
* **Highest Response Ratio Next (HRRN):**
* **Mô tả:**
* Cơ chế thực thi:
* 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:
* **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:** Phải tính RR (Response ratio) để xác định process được chạy
* **Multilevel Queue Scheduling (MQ):**
* **Mô tả:**
* Cơ chế thực thi:
* 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.
* **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 (MFQ):**
* **Mô tả:**
* Cơ chế thực thi:
* (Tương tự Multilevel Feedback Queue).
* Điểm khác biệt: Cho phép process nhảy từ queue này đến queue khác.
* **Ưu điểm:**
* Một process đã chờ quá lâu ở một hàng đợi có độ ưu tiên thấp có thể được chuyển đến hàng đợi có độ ưu tiên cao hơn (cơ chế aging)
* Cho phép process di chuyển một cách thích hợp giữa các hàng đợi khác nhau.
* Thuật toán chung nhất, có thể được thiết kế để phù hợp với các hệ thống khác biệt.
* **Nhược điểm**: Phải giải quyết các câu hỏi:
* Số lượng hàng đợi bao nhiêu là thích hợp?
* Dùng giải thuật định thời nào ở mỗi hàng đợi?
* Làm sao để xác định thời điểm cần chuyển một process đến hàng đợi cao hơn hoặc thấp hơn?
* Khi process yêu cầu được xử lý thì đưa vào hàng đợi nào là hợp lý nhất?

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

* 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).
* Thực hiện cân bằng tải khi: Một bộ xử lý có quá nhiều tải, trong khi các bộ xử lý khác rỗi => Cần đảm bảo các bộ xử lý đều được sử dụng hiệu quả.

**Câu 8: 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?**

* **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 trưng 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.

**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:**

* 1. **FCFS**

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

0 8 27 30 36 46

* Thời gian chờ trung bình: (0 + 6 + 23 + 25 + 29)/5 = 16.6
* Thời gian đáp ứng trung bình: (0 + 6 + 23 + 25 + 29)/5 = 16.6
* Thời gian hoàn thành trung bình: (8 + 25 + 26 + 31 + 39)/5 = 25.8
  1. **SJF preemptive**

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

0 4 7 11 17 27 46

* Thời gian chờ trung bình: (3 + 25 + 0 + 6 + 10)/5 = 8.8
* Thời gian đáp ứng trung bình: (0 + 25 + 0 + 6 + 10)/5 = 8.2
* Thời gian hoàn thành trung bình: (11 + 44 + 3 + 12 + 20)/5 = 18

**c. RR với quantum time = 6**

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

0 6 12 15 21 23 29 35 39 46

* Thời gian chờ trung bình: (15 + 25 + 8 + 10 + 22)/5 = 16
* Thời gian đáp ứng trung bình: (0 + 4 + 8 + 10 + 16)/5 = 7.6
* Thời gian hoàn thành trung bình: (23 + 44 + 11 + 16 + 32)/5 = 25.2

**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 |

**a. FCFS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P5 | P6 | P3 | P2 | P4 |

0 20 55 105 130 155 170

* Thời gian chờ trung bình: (0 +105 + 85 + 120 + 10 + 40)/6 = 60
* Thời gian đáp ứng trung bình: (0 +105 + 85 + 120 + 10 + 40)/6 = 60
* Thời gian hoàn thành trung bình: (20 + 130 + 110 + 135 + 45 + 90)/6 = 88.333

**b. SJF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P3 | P4 | P2 | P5 | P6 |

0 20 45 60 85 120 170

* Thời gian chờ trung bình: (0 + 35 + 0 + 10 + 75 + 105)/6 = 37.5
* Thời gian đáp ứng trung bình: (0 + 35 + 0 + 10 + 75 + 105)/6 = 37.5
* Thời gian hoàn thành trung bình: (20 + 60 + 25 + 25 + 110 + 155)/6 = 65.8(3)

**c. SRTF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P3 | P4 | P2 | P5 | P6 |

0 20 45 60 85 120 170

* Thời gian chờ trung bình: (0 + 35 + 0 + 10 + 75 + 105)/6 = 37.5
* Thời gian đáp ứng trung bình: (0 + 35 + 0 + 10 + 75 + 105)/6 = 37.5
* Thời gian hoàn thành trung bình: (20 + 60 + 25 + 25 + 110 + 155)/6 = 65.83333

**d. Priority - Pre**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P1 | P5 | P6 | P3 | P1 | P2 | P4 |

0 10 45 95 120 130 155 170

* Thời gian chờ trung bình: (110 + 105 + 75 + 120 + 0 + 30)/6 = 73.3333
* Thời gian đáp ứng trung bình: (0 + 105 + 75 + 120 + 0 + 30)/6 = 55
* Thời gian hoàn thành trung bình: (130 + 130 + 100 + 135 + 35 + 80)/6 = 101.666

**e. RR (10)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P5 | P1 | P6 | P3 | P5 | P2 | P4 | P6 | P3 | P5 | P2 | P4 | P6 | P3 | P5 | P2 | P6 |

0 10 20 30 40 50 60 70 80 90 100 110 120 125 135 140 145 150 170

* Thời gian chờ trung bình: (10 + 100 + 95 + 75 + 100 + 105)/6 = 80.8(3)
* Thời gian đáp ứng trung bình: (0 + 35 + 20 + 35 + 0 + 15)/6 = 17.5
* Thời gian hoàn thành trung bình: (30 + 125 + 120 + 90 + 135 + 155)/6 = 109.1(6)

**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:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

* 1. **SJF Preemptive**

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

0 1 4 5 7 12 21

* Thời gian chờ trung bình: (11 + 0 + 3 + 1 + 3)/5 = 3.6
* Thời gian đáp ứng trung bình: (0 + 0 + 3 + 1 + 3)/5 = 1.4
* Thời gian hoàn thành trung bình: (21 + 3 + 5 + 2 + 8)/5 = 7.8

* 1. **RR với quantum time = 2**

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

0 2 4 6 8 9 11 12 14 16 18 19 21

* Thời gian chờ trung bình: (11 + 8 + 2 + 5 + 10)/5 = 7.2
* Thời gian đáp ứng trung bình: (0 + 1 + 2 + 5 + 5)/5 = 2.6
* Thời gian hoàn thành trung bình: (21 + 11 + 4 + 6 + 15)/5 = 11.4

**c. Preemptive Priority (độ ưu tiên 1 > 2 > ...)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 = 4.6
* Thời gian đáp ứng trung bình: (0 + 0 + 0 + 3 + 12)/5 = 3
* 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:**

* 1. **FCFS**

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

0 10 39 42 49 61

* Thời gian chờ trung bình: (0 + 8 + 35 + 37 + 42)/5 = 24.4
* Thời gian đáp ứng trung bình: (0 + 8 + 35 + 37 + 42)/5 = 24.4
* Thời gian hoàn thành trung bình: (10 + 37 + 38 + 44 + 54)/5 = 36.6

* 1. **SJF preemptive**

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

0 4 7 13 20 32 61

* Thời gian chờ trung bình: (3 + 30 + 0 + 8 + 13)/5 = 10.8
* Thời gian đáp ứng trung bình: (0 + 30 + 0 + 8 + 13)/5 = 10.2
* Thời gian hoàn thành trung bình: (13 + 59 + 3 + 15 + 25)/5 = 23
  1. **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 + 30 + 16 + 18 + 33)/5 = 19.4
* Thời gian đáp ứng trung bình: (0 + 8 + 16 + 18 + 23)/5 = 13
* Thời gian hoàn thành trung bình: (10 + 59 + 19 + 25 + 45)/5 = 31.6